Oracle® Invoice Matching

Operations Guide

Release 16.0.2.1

F13389-02

September 2021

ORACLE

Oracle Retail Invoice Matching Operations Guide, Release 16.0.2.1

F13389-02

Primary Author:

reconfigurations, reassembly or reverse assembly, re-engineering or reverse engineering and recompilations or reverse compilations of the VAR Applications or any derivatives of the VAR Applications. You acknowledge that it shall be a breach of the agreement to utilize the relationship, and/or confidential information of the VAR Applications for purposes of competitive discovery.

The VAR Applications contain trade secrets of Oracle and Oracle's licensors and Customer shall not attempt, cause, or permit the alteration, decompilation, reverse engineering, disassembly or other reduction of the VAR Applications to a human perceivable form. Oracle reserves the right to replace, with functional equivalent software, any of the VAR Applications in future releases of the applicable program.

Contents

3 Backend System Administration and Configuration

8 Integration

9 Technical Design

10 Oracle E-Business Suite Financials Integration using Oracle Retail Financial Integration

13 Batch Processes

Send Us Your Comments

Oracle Retail Invoice Matching Operations Guide, Release 16.0.2.1

Oracle welcomes customer comments and suggestions on the quality and usefulness of this document.

Your feedback is important, and helps us to best meet your needs as a user of our products. For example:

- Are the implementation steps correct and complete?
- Did you understand the context of the procedures?
- Did you find any errors in the information?
- Does the structure of the information help you with your tasks?
- Do you need different information or graphics? If so, where, and in what format?
- Are the examples correct? Do you need more examples?

If you find any errors or have any other suggestions for improvement, then please tell us your name, the name of the company who has licensed our products, the title and part number of the documentation and the chapter, section, and page number (if available).

> **Note:** Before sending us your comments, you might like to check that you have the latest version of the document and if any concerns are already addressed. To do this, access the new Applications Release Online Documentation CD available on My Oracle Support and www.oracle.com. It contains the most current Documentation Library plus all documents revised or released recently.

Send your comments to us using the electronic mail address: retail-doc_us@oracle.com

Please give your name, address, electronic mail address, and telephone number (optional).

If you need assistance with Oracle software, then please contact your support representative or Oracle Support Services.

If you require training or instruction in using Oracle software, then please contact your Oracle local office and inquire about our Oracle University offerings. A list of Oracle offices is available on our Web site at www.oracle.com.

Preface

This Operations Guide provides critical information about the processing and operating details of Document Template, including the following::

- System configuration settings
- Technical architecture
- Functional integration dataflow across the enterprise
- Batch processing

Audience

This guide is for:

- Systems administration and operations personnel
- Systems analysts
- Integrators and implementers
- Business analysts who need information about Product processes and interfaces

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Related Documents

For more information, see the following documents in the Oracle Retail 16.0 documentation set:

- *Oracle Retail Invoice Matching Release Notes*
- *Oracle Retail Invoice Matching Installation Guide*

- *Oracle Retail Invoice Matching User Guide*
- *Oracle Retail Invoice Matching Data Model*
- *Oracle Retail Merchandising Batch Schedule*
- *Oracle Retail Merchandising Implementation Guide*
- *Oracle Retail Merchandising Security Guide*

Customer Support

To contact Oracle Customer Support, access My Oracle Support at the following URL:

https://support.oracle.com

When contacting Customer Support, please provide the following:

- Product version and program/module name
- Functional and technical description of the problem (include business impact)
- Detailed step-by-step instructions to re-create
- Exact error message received
- Screen shots of each step you take

Review Patch Documentation

When you install the application for the first time, you install either a base release (for example, 16.0) or a later patch release (for example, 16.0.2). If you are installing the base release and additional patch releases, read the documentation for all releases that have occurred since the base release before you begin installation. Documentation for patch releases can contain critical information related to the base release, as well as information about code changes since the base release.

Improved Process for Oracle Retail Documentation Corrections

To more quickly address critical corrections to Oracle Retail documentation content, Oracle Retail documentation may be republished whenever a critical correction is needed. For critical corrections, the republication of an Oracle Retail document may at times not be attached to a numbered software release; instead, the Oracle Retail document will simply be replaced on the Oracle Technology Network Web site, or, in the case of Data Models, to the applicable My Oracle Support Documentation container where they reside.

This process will prevent delays in making critical corrections available to customers. For the customer, it means that before you begin installation, you must verify that you have the most recent version of the Oracle Retail documentation set. Oracle Retail documentation is available on the Oracle Technology Network at the following URL:

http://www.oracle.com/technetwork/documentation/oracle-retail-100266.html

An updated version of the applicable Oracle Retail document is indicated by Oracle part number, as well as print date (month and year). An updated version uses the same part number, with a higher-numbered suffix. For example, part number E123456-02 is an updated version of a document with part number E123456-01.

If a more recent version of a document is available, that version supersedes all previous versions.

Oracle Retail Documentation on the Oracle Technology Network

Documentation is packaged with each Oracle Retail product release. Oracle Retail product documentation is also available on the following Web site:

http://www.oracle.com/technetwork/documentation/oracle-retail-100266.html

(Data Model documents are not available through Oracle Technology Network. These documents are packaged with released code, or you can obtain them through My Oracle Support.)

Documentation should be available on this Web site within a month after a product release.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
monospace	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1
Introduction

Oracle Retail Invoice Matching (ReIM) provides a critical control function to verify invoices against corresponding merchandise purchase receipts prior to payment of the supplier invoice. ReIM naturally complements the Oracle Retail Merchandising System (RMS), which supports ordering, receiving, and other inventory management functions in the purchasing cycle.

ReIM accurately and efficiently verifies supplier invoices against corresponding receipt data. When total invoice cost and quantity is supported by one or more receipts (that is, the quantity received in the system, valued at the negotiated purchase order cost) within pre-defined tolerances, the invoice is verified (or matched) and is ready for payment. Where differences exist between invoice and receipt, a dialog supports the resolution process. Invoices with resolved discrepancies can be paid. Invoices verified for payment are staged in a table for a retailer to extract to their accounts payable and general ledger solutions.

ReIM is designed as a standalone application, with logic built in to reference any merchandising system. However, integration between ReIM and RMS is very robust and offers a compelling business case to the retailer.

What is Retail Invoice Matching?

Invoice matching describes a control procedure designed to ensure the retailer pays the negotiated cost for actual quantities received. Invoice verification or matching is a fundamental and critical control procedure for every retailer.

ReIM is designed to support the invoice verification process with accuracy and efficiency, focusing resources on exception management. ReIM accepts electronic invoice data uploads (EDI), and provides for rapid on-line summary entry of invoices. ReIM supports automated and on-line processes allowing one or more invoices to be matched against one or more receipts. When an invoice cost and quantities are matched within tolerance, it is ready for payment and staged to a table to allow a retailer to extract to their accounts payable solution.

If a cost or quantity difference between the invoice and receipts is outside tolerance, a discrepancy is recognized and must be resolved. A flexible resolution process allows discrepancies to be directed to the most appropriate user group for disposition. Reviewers are empowered to assign one or more reason codes that they are authorized to use, to resolve the discrepancy.

Each reason code is associated to a type of action (for example, create charge back or receiver cost adjustment). Many reason codes may be associated with a particular action type, allowing for more granular reporting, and so on. Actions drive document creation and EDI downloads to suppliers, inventory adjustments, and accounting

activities. Actions also allow the invoice to be extracted by the retailer and posted for payment.

ReIM is highly integrated with RMS to drive efficiency, lower maintenance costs and improve control. ReIM integration provides access to the following data and more:

- RMS foundation data (organizational and merchandising hierarchies, supplier data, currency, exchange rates, and so on)
- Receipts tables and receiver adjustments
- Self-billing transactions (consignment purchases, direct store deliveries, and so on)
- RTV billings
- Deals and rebate bill-backs

Other functionality within ReIM supports credit note matching against credit note requests (issued in resolution of invoice discrepancies, as well as for RTVs and so on), supplier-disputed debit memos, best terms and terms date processing, flexible tolerance definition dialog, and so on.

Oracle Retail-Based Enterprises

Although ReIM has been developed as a stand-alone product, the most efficient implementation would be as part of the Oracle Retail product suite. This integration provides the following important benefits:

- The number of interface points that need to be maintained is minimized.
- The amount of redundant data and processes within the retail organization is limited.
- Future enhancements allow for greater extensibility into the retail enterprise.
- Delays in product introductions can be minimized.

Technical Architecture Overview

The Java architecture is built upon a layering model. That is, layers of the application communicate with one another through an established hierarchy and are able to communicate only with neighboring layers.

For more information, see "Chapter 2, "Technical Architecture".

2

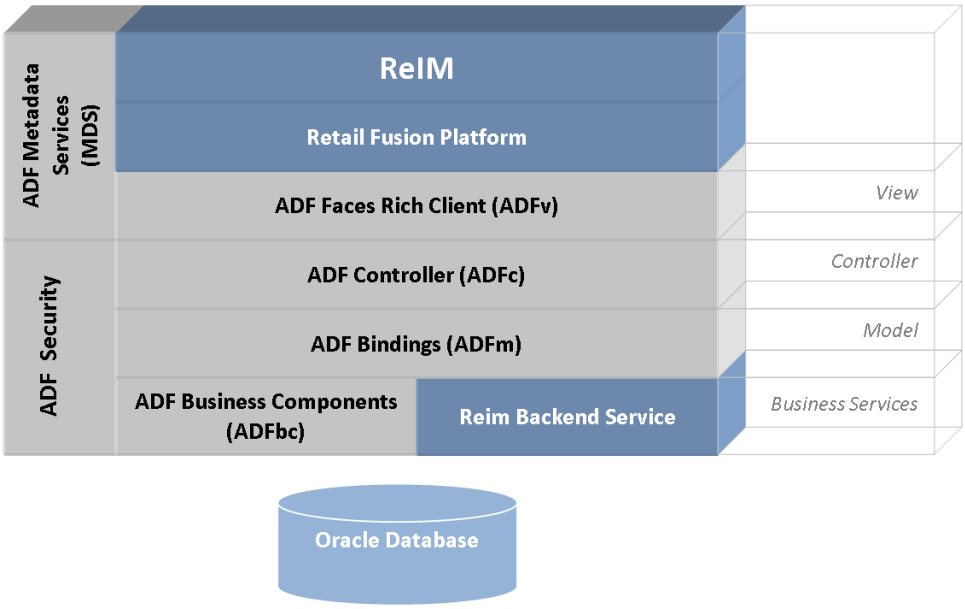
Technical Architecture

This chapter describes the overall software architecture for Oracle Retail Invoice Matching. The chapter provides a high-level discussion of the general structure of the system, including the various layers of Java code.

Overview

Retail Applications are based on the Oracle Application Development Framework (ADF). The following diagram shows the key components that make up the architecture of Retail Applications.

Figure 2–1 Oracle Retail Invoice Matching n-tier Architecture



Oracle Application Development Framework (ADF)

Oracle Application Development Framework (ADF) supports organizations in building cutting-edge rich enterprise business applications that can be customized and personalized in all dimensions. Customizations are global changes, visible to all users that are performed by an administrator. Personalization are user-made changes that are only visible to the person making the change.

ADF is based on the Java Enterprise Edition platform.

Model-View-Controller (MVC) Architectural Pattern

Applications built using ADF follow a Model-View-Controller (MVC) architectural pattern. The goal of the MVC pattern is to clearly separate the application's functionality into a set of cooperating components.

ADF provides a set of components that realize the goals of each part of MVC pattern.

- Model is realized by the ADF Bindings Layer.
- Controller is realized by the ADF Controller Layer.
- View is realized by the ADF Faces Layer.
- ADF Business components and other backend components that sit below the Model layer are called Business Services.

ADF Security

The ADF security layer provides the following:

- Standards based (Oracle Platform Security Services (OPSS)) security framework with default roles and permissions.
- Tools to generate file-based identity store (for both Oracle Internet Directory and AD) based on the framework.
- Tools to migrate file-based security store in to database for QA and production environments.
- Reference implementation for clients to manage the security based on their business needs.
- OPSS-based batch security framework (Retail Fusion Platform).
- Tools/documentation to implement centralized logout in Single Sign-On (SSO) (Oracle Access Management (OAM)) environments.

ADF View (ADFv)

The View layer provides the user interface to the application. The view layer uses HTML, rich Java components or XML and its variations to render the user interface. JSF based tag libraries are used for displaying the UI.

ADF Controller (ADFc)

The ADF Controller layer controls the application's flow. Web based applications are composed of multiple web pages with dynamic content. The controller layer manages the flow between these pages. Different models can be used when building this later. The most prominent architecture for Java-based web applications relies on a servlet that acts as the controller. The Apache Jakarta Struts controller, an open source framework controller, is the de facto standard for Java-based web systems. Oracle ADF uses the Struts controller to manage the flow of web applications.

ADF Business Components (ADFbc)

The business service layer manages the interaction with a data persistence layer. It provides services as data persistence, object/relational mapping, transaction management and business logic execution.

Business Components easily map the database object and extend it with business logic, validation and so on.

The idea behind Business Components is to abstract the data layer from the view layer. This is a key concept in the MVC pattern. Business Components will expose the interface to the view layer by using an application module that contains View Object. Those view objects contain a specific usage of the data layer.

ADF Business Components implements the business service through the following set of cooperating components:

- Entity object – An entity object represents a row in a database table and simplifies modifying its data by handling all data manipulation language (DML) operations for you. It can encapsulate business logic for the row to ensure that your business rules are consistently enforced. You associate an entity object with others to reflect relationships in the underlying database schema to create a layer of business domain objects to reuse in multiple applications.
- View object – A view object represents a SQL query. You use the full power of the familiar SQL language to join, filter, sort, and aggregate data into exactly the shape required by the end-user task. This includes the ability to link a view object with others to create master-detail hierarchies of any complexity. When end users modify data in the user interface, view objects collaborate with entity objects to consistently validate and save the changes.
- Application module – An application module is the transactional component that UI clients use to work with application data. It defines an updatable data model and top-level procedures and functions (called service methods) related to a logical unit of work related to an end-user task.

ADF Model (ADFm)

This component acts as the connector between the view and business logic layers.

The Model layer connects the Business Services to the objects that use them in the other layers. Oracle ADF provides a Model layer implementation that sits on top of Business Services, providing a single interface that can be used to access any type of Business Services.

Developers get the same development experience when binding any type of Business Service layer implementation to the view and Controller layers. The Model layer in Oracle ADF served as the basis for JSR 227, A Standard Data binding & Data Access Facility for J2EE.

Oracle Metadata Services (MDS)

The ability of an application to adapt to changes is a necessity that needs to be considered in the application design and that should drive the selection of the development platform and architecture. Flexible business applications must be able to adapt to organizational changes, different end user preferences and changes in the supported business are required.

MDS is the customization and personalization framework integral to Oracle Fusion Middleware and a key differentiator of the Oracle development platform. MDS provides a repository for storing metadata for applications, such as customizations and persisted personalization files and configurations.

Retail Applications allow the following through MDS:

- Personalization of saved searches through MDS.
- Implicit personalization of few ADF UI attributes.

Retail Fusion Platform

The Retail Fusion Platform (commonly referred to as Platform) is a collection of common, reusable software components that serve as foundation for building Oracle Retail's next generation ADF-based applications. The Platform imposes standards and patterns along with a consistent look and feel for Oracle Retail's ADF applications.

RelM Backend Service

Reim Backend Services are collective Business Legacy components that are reused in the new ADF Version of Reim.

Data Access Patterns

Database interaction between the middle tier and the database is done using the industry standard Java Database Connectivity Protocol (JDBC). JDBC facilitates the communication between a Java application and a relational database.

Database Access Using ADFbc

JDBC is engrained within Oracle ADF Business Components as the primary mechanism for its interaction between the middle tier and the database. SQL is realized within ADF business components to facilitate create, read, update and delete (CRUD) actions.

Connection Pooling

When the application 'disconnects' a connection, the connection is saved into a pool instead of being actually disconnected. A standard connection pooling technique, this saved connection enables Retail Applications to reuse the existing connection from a pool. In other words, the application does not have to complete the connection process for each subsequent connection.

Data Storage

The Oracle Database realizes the database tier in a Retail Application's architecture. It is the application's storage platform, containing the physical data (user and system) used throughout the application. The database tier is only intended to handle the storage and retrieval of information and is not involved in the manipulation or in the delivery of the data. This tier responds to queries; it does not initiate them.

Accessing Merchandising System Data in Real Time

The data that Retail Application utilizes is located in both application-specific tables and merchandising system (RMS, for example) tables. Because Retail Applications share the same schema as the merchandising system (RMS, for example), the application is able to interact with the merchandising system's data directly, in real time.

3

Backend System Administration and Configuration

This chapter of the operations guide is intended for administrators who provide support and monitor the running system.

The content in this chapter is not procedural, but is meant to provide descriptive overviews of the key system parameters that establish the ReIM environment.

See the *Oracle Retail Invoice Matching Installation Guide* for hardware and software requirements. Also see Oracle Retail application software compatibility information.

System Assumptions

- Unit of Measure

  For invoices sent from RMS with quantities representing weight rather than number of eaches, ReIM converts the unit of measure (UOM) on the receipt to the UOM on the invoice.

- ReIM uses non-merchandise codes defined on the RMS table NON_MERCH_CODE_HEAD. The form that allows users to enter non-merchandise codes in RMS is not available when the RMS invoice match indicator (SYSTEM_OPTIONS.REIM_IND) is set to no. Instead, non-merchandise codes should be added to the NON_MERCH_CODE_HEAD table using the database.

- Supplier options

  All suppliers must have options defined for their invoices to be processed by the system, and the terms defined for those suppliers must be completely updated in RMS. To support the use of suppliers in ReIM, terms must have the following properties on the TERMS_DETAIL table:

  – ENABLED_FLAG is set to Y.

  – START_DATE_ACTIVE must be defined.

  – END_DATE_ACTIVE must be defined.

- GL account maintenance

  All reason codes, non-merchandise codes, and basic transactions must be mapped through GL account maintenance to support posting to the retailer's financial solution. Transactions are posted to a staging table in ReIM, the extract to update the accounts payable/financial solution is the retailer's responsibility.

- TAX

If TAX is turned on, the retailer must have TAX regions, TAX items, and TAX codes set up in the merchandising system (such as RMS) to support validation of invoiced TAX charges. Verify the following values on the IM_SYSTEM_OPTIONS table:

Note: The values below should not be changed after initial setup. Changing them can cause errors in the system.

- NUM_TAX_ALLOW is set to S (single) TAX, N (no) TAX.

  TAX_VALIDATION_TYPE is set to RECON (Reconcile TAX), VENDR (Always Use Vendor TAX), or RETLR (Always use Retail TAX).
- The DEFAULT_TAX_HEADER is set to Y or N.
- TAX_DOCUMENT_CREATION_LVL is set to ITEM or FULL_INVOICE.

reim.properties File

The property file has only technical configurations that are specific to the particular deployment. All the business related configurations are located within the system option scope.

The following properties are configurable

- #help settings
  - help.server.url - Help server URL
  - reim.help.library - path to the ReIM specific help deck
  - reim.release.version - help version
  - reim.help.url.suffix -help URL suffix identifying specific document within the library
  - reim.help.default.url.suffix -default page within the document
- #JMS settings
  - reim.jms.connection.factory.name -name of the JMS factory within JNDI tree on WL used for spreadsheet induction
  - reim.jms.queue.doc.induction.name - name of the JMS queue within JNDI tree on WL used for spreadsheet induction
  - reim.jms.application.code- Application code identifier used for JMS publishing
- #Notification settings
  - reim.jdbc.raf.async.task- name of the data source within JNDI tree on WL used for async publishing

Logging Configuration

Oracle Retail Invoice Matching utilizes the industry-standard Apache Log4j logging framework to log system messages and exceptions. This framework is embedded in the application code to allow for configurable logging to suit the needs of the retailer.

Please note that Log4j is used for batch clients only. Server logging is done using standard WbLogic logging infrostracture.

Log4J Conventions

The Log4j API system utilizes three main configurable entities:

- Loggers
- Appenders
- Layouts

Loggers are responsible for defining exactly what gets logged. Typically, loggers define a specific level of detail (the log level) for a specific java package name as well as an appender the logger is assigned to. These criteria are then delegated to the appropriate appender for the specific logger. A single logger can be assigned to multiple appenders.

Appenders are used to dictate where logged content is directed to for a given logger. For example, the retailer may wish to configure a log appender to publish a log to a database table, a flat file, or an e-mail address. For each of these options, a separate appender would be defined and assigned to a specific logger.

Layouts are leveraged by the appender to dictate the exact content of the log message. Relevant information may include: date, time, and origin of the error message. These values can all be configured through the log layout.

Log4J Properties

The log4j.properties file holds all of the information relevant to logging for batch clients.

Table 3–1 Log4J Parameters

Parameter	Description
log4j.appender.BATCHFILE	Type of file appender.
log4j.appender.BATCHFILE.File	Path to the log file where batches will write log entries.
log4j.appender.BATCHFILE.DatePattern	Date pattern for log file.
log4j.appender.BATCHFILE.Append	Indicator if the log should append entries to the existing file.
log4j.appender.BATCHFILE.Threshold	Logging threshold level.
log4j.appender.BATCHFILE.layout	Pattern layout for log entry.
log4j.appender.BATCHFILE.layout.ConversionPattern	Pattern for log entry.

Internationalization

Internationalization is the process of creating software that is able to be translated more easily. Changes to the code are not specific to any particular market. ReIM has been internationalized to support multiple languages.

This section describes configuration settings and features of the software that ensure that the base application can handle multiple languages.

Translation

Translation is the process of interpreting and adapting text from one language into another. Although the code itself is not translated, components of the application that are translated may include the following:

- Graphical user interface (GUI)
- Error messages

The following components are not translated:

- Documentation (online help, release notes, installation guide, user guide, operations guide)
- Batch programs and messages
- Log files
- Configuration tools
- Reports
- Demonstration data
- Training materials

The user interface for ReIM has been translated into:

- Arabic
- Chinese (simplified)
- Chinese (traditional)
- Croatian
- Dutch
- French
- German
- Greek
- Hungarian
- Italian
- Japanese
- Korean
- Polish
- Portuguese (Brazilian)
- Russian
- Spanish
- Swedish
- Turkish

Setting the User Language

To set the language ReIM displays in, set the user language. Choose Preferences from the drop-down menu under the user name, then choose Language in the taskbar. There are two language settings you can choose:

- Default: This will permanently change the language for this user. All subsequent sessions will be in this language.
- Current Session: This will change the language only for this session. It will revert back to the Default the next time this user logs in.
- When finished making your selections, click the Save button.

Setting Date, Time, and Number Formats

ReIM allows the user to see dates and times in a format appropriate for their own locale, regardless what the data is stored in. Set the language ReIM displays in, set the user language. Choose Preferences from the drop-down menu under the user name, then choose Regional in the taskbar. There are multiple settings you can choose:

- Territory: Choose a territory from this list. ReIM will automatically pick Date, Time, and Number formats appropriate for that territory.
- Date Format: Choose an option from this menu to select a date format that is different from the default for the selected Territory.
- Time Format: Choose an option from this menu to select a time format that is different from the default for the selected Territory.
- Number Format: Choose an option from this menu to select a number format that is different from the default for the selected Territory.
- Time Zone: Choose an option from this menu to select your time zone.

When finished making your selections, click the **Save** button.

Translations

Most user interface and message translations are stored in xlf files. When you select a different language from the Preferences screen, ReIM will choose the correct xlf file for that language.

Some translations for some drop-down menus are stored on four database tables. The tables are RTC_LOOKUP_VALUES_TL, RTC_LOOKUP_TYPES_TL, RAF_FACET_ATTRIBUTE_CFG_TL, and RAF_NOTIFICATION_TYPE_TL. These tables are multilingual; all languages of these strings (English as well as all translations) are stored in the same place.

ReIMResources.properties

This file contains a key value pair for some of the labels visible through the GUI at run time. Text labels and error messages have been identified, separated from the core source code, and placed into the properties file. The contents of the file can be used for retailer-specific configuration purposes (such as for the creation of custom labels or error messages). Some other sources of translatable data is used within the system such as XLIF files and database tables.

batch.properties

This file contains security information about batch clients.

Table 3–2 batch.properties Parameters

Parameter	Description
providerUrl	URL of the server JNDI.
csm.wallet.partition.name	Name of the wallet partition where the batch credentials are stored.
csm.wallet.path	Path of the wallet where the batch credentials are stored

Managing the Asynchronous Processes

Overview of Asynchronous Processes in Retail Applications

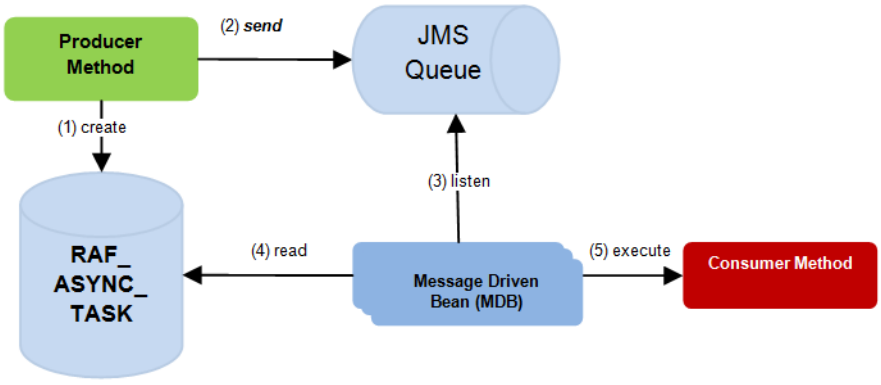
Applications may need to execute operations that can take significant time to complete.

Executing these operations in the application's own thread will cause the application to block other operations until those long-running operations finish.

In contrast, being able to assign the long-running operations to background threads allows the foreground thread to remain active and service further user requests. This is also described as launching those operations asynchronously.

For example, in the Retail Reim application, when an invoice induction request is submitted for approval, the system needs to perform comprehensive validation to ensure that the request is valid. The validation should be performed asynchronously. The user can proceed with performing other operations within the system. Once the background validation is complete, the user is notified.

Retail Applications achieve asynchronous processing using Java Messaging Service (JMS). When a user, for instance, through the application UI screen, submits a business entity for processing asynchronously, the application goes through a process depicted in the diagram below:



1. The Retail Application's producer process executes to collect the user request for asynchronous processing and stores that information into a database table called RAF_ASYNC_TASK. This table serves as a log of asynchronous processing requests as well as storage for any kind of context information in order to complete the processing. Example: The unique identifier of the business object for each invoice induction request.

2. The producer method creates a queue message and sends it to a JMS Queue.
3. One or more instances of Message Driven Beans (MDB) are configured to listen to the JMS Queue.
4. When messages are detected on the queue, the MDBs are dispatched to execute a task asynchronously. The MDB will read context information about the task from the task table.
5. The MDB will execute the required processing for the task.

Configuring JMS Resources for Asynchronous Processing

The following resources in the WebLogic Administrator Console application should be configured in order to allow asynchronous processing to function in a Retail Application:

WebLogic Server Resource	Required Names/References	Configuration Notes
JMS Server	Any name can be used.	None. Defaults provided by WebLogic acceptable.
JMS Module	Any name can be used.	None. Defaults provided by WebLogic acceptable.
JMS Queue	Name: InvoiceInductionQueue JNDI: jndi/InvoiceInductionQueue	▪ Create the queue under the JMS Module. ▪ Associate a sub-deployment for the queue. If none exists, create it. The sub-deployment must be the same as the JMS Queue Connection Factory's. ▪ Redelivery Limit must be set to zero (0). ▪ Error Destination must be set to NONE.
JMS Queue Connection Factory	Name: InvoiceInductionCF JNDI: jndi/InvoiceInductionCF	▪ Create under the same JMS Module as the queue. ▪ Associate a sub-deployment for the queue connection factory. If none exists, create it. The sub-deployment must be the same as the JMS Queue's. ▪ Maximum Messages per Session must be set to one (1). ▪ XA Connection Factory Enabled must be true.

Please refer to WebLogic documentation found in **http://www.oracle.com/technetwork/middleware/weblogic/documentation/index.html** for detailed information about how to configure the above resources.

Monitoring Asynchronous Tasks via the RAF_ASYNC_TASK Table

The RAF_ASYNC_TASK table can be used to view error codes for failures, and to monitor any failures requiring triage from production support team. As described in the section, Overview of Asynchronous Processes in Retail Applications, each asynchronous processing request is logged as a row in the RAF_ASYNC_TASKS table.

As the task is sent to the queue, picked up and completed by the consuming message driven bean, the row in this table is updated with status information.

RAF_ASYNC_TASK Column	Description
ASYNC_TASK_ID	A unique identifier for each row in this table. Each asynchronous task is given a unique ASYNC_TASK_ID.
APPLICATION_CODE	A code representing the Retail Application that generated the asynchronous task.
TASK_DESC	A short description of the asynchronous task.
TASK_CONTEXT	A string containing information needed to process the asynchronous task. Usually indicates the ID of the business entity to be processed (example: the ID of the request to be processed)
STATUS	The status of the asynchronous task. ■ NEW - The task is ready to be asynchronously processed. ■ IN-PROGRESS - The task is currently being processed by the application. ■ SUCCESS - Processing for the task completed successful. ■ FAILED - Processing for the task completed with failures. Additional information about the failure can be found in the PROCESS_ERROR_TXT column and/or the server logs.
TASK_COMMAND_CLASS_NAME	A fully-qualified Java class name that contains the business logic for completing the task.
PUBLISH_TIMESTAMP	The time and date when the asynchronous task was published. That is, when the status was changed to NEW.
PROCESS_START_TIMESTAMP	The time and date when processing of the asynchronous task started. That is, when the status was changed to IN-PROGRESS.
PROCESS_END_TIMESTAMP	The time and date when the processing of the asynchronous task ended with either a SUCCESS or FAILED status.
PROCESS_ERROR_TXT	When the status is FAILED, this column would contain information about the failure. The server logs may need to be inspected for more detailed information about the cause of the processing failure.
CREATED_BY	Audit field pertaining to the logged in application user that requested the task.
CREATE_DATE	Audit field pertaining to when the row was created.
LAST_UPDATED_BY	Audit field pertaining to the user that last updated the row.
LAST_UPDATE_DATE	Audit field pertaining to the last date/time when the row was updated.

Purging the Asynchronous Tasks Table

As previously mentioned, each asynchronous task is logged as a row in the RAF_ASYNC_TASK table. Overtime, this table can grow significantly, which can degrade performance of the asynchronous task mechanism.

It is recommended that retailers periodically purge this table.

Retail Applications provide a simple PL/SQL function to purge contents of the RAF_ASYNC_TASK table based on retention period (in days).

```
declare
   retval number(10);
begin
   -- remove rows older than 5 days
   retval := raf_async_task_pkg.delete_async_task(5);
end;
/
```

Managing the Notifications Feature

This section covers the following topics:

- Overview of Notifications in Retail Applications
- Purging the Notifications Table

Overview of Notifications in Retail Applications

Retail Applications provide the ability to notify authenticated users in the application when business events occur. An example of a business event is when a background asynchronous task such as an approval of an allocation or a validation of a price change has completed. Typically, the user is expected to take action on these notifications.

The notification icon on the Retail Application's navigation pane displays the count of unread notifications for the user. When a notification is generated, this count is incremented immediately and thereafter refreshed at regular configurable intervals.

The sliding sidebar menu expands when the user clicks on the notification icon. The expanded sidebar menu will show the latest notifications. Clicking on "See All" will show all the notifications in a new tab.

Clicking on the link for each launchable notification opens the specific UI flow that will allow the user to address any pending action for the notification.

Figure 3–1 Sliding Sidebar showing the Notification Count

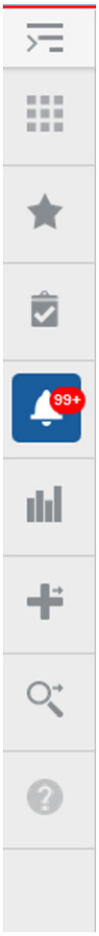
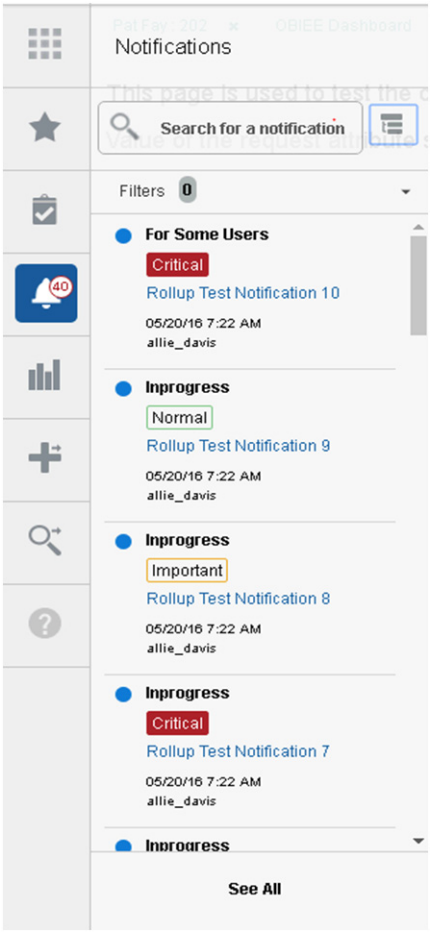


Figure 3–2 Notifications Sidebar Menu



Purging the Notifications Table

Notifications are represented as rows in the table RAF_NOTIFICATION. Over a period of time, depending on how notifications are being generated by the application, the size of this table can grow continually, potentially degrading performance of the notifications feature.

It is recommended that retailers purge this table periodically. You can do this via the Purge feature available in the Retail Application Administration Console (RAAC). For more details on the Manage Notifications feature in RAAC, please refer to the Oracle Retail Application Administration Console User Guide.

You could also setup a batch job that runs periodically, that invokes the pl-sql as shown below:

```
declare
   retval number(10);
begin
   -- remove Notifications from the Reim application that have exceeded       .
-- their retention duration.
   retval := raf_notification_task_pkg. DEL_NOTIF_PAST_RETENTION('REIM');
end;
```

Notification ReST Services

ReST Endpoints have been exposed by the Notifications framework to ease the integration concerns with disparate applications.

Enabling Mobile Push Notifications

Users of the mobile version of the Retail Application can receive notifications in their mobile devices using push notifications.

Push notifications for Mobile Retail Applications need to be enabled by the Retailer. This section describes the specific steps to enable this feature.

Note that push notification is a native feature in modern mobile devices so the steps to enable this service on different platforms will be different.

Prerequisite: Android Google Cloud Messaging Registration

Using push notifications on Android requires retailers to register the mobile Retail Application with Google Cloud Messaging (GCM). An API key and sender ID will be generated from the account created during registration. Refer to Google's developer site for instructions on how to create an account and generate these credentials.

Server Configuration

Once credentials to the specific platforms are available, retailers need to add these to the server.

For Apple Push Notifications, retailers must import the key and certificate from Apple into weblogic. In order to do this, a keystore (JKS) file must be created using the Java keystore utility. The key and certificate can then be imported into the created keystore file. Retailers can refer to Oracle Java SE documentation for details on the keystore utility.

Once created, the keystore can be uploaded into the weblogic system app stripe. Retailers can use the weblogic scripting tool to run a Python script similar to the one shown below. The example shows that the keystore is named RetailAppsPushServices.

```
user = "admin username"
password = "admin password"
url = "[weblogic server]:[port]"
keypass = "keystore password"
keypath = "keystore filepath"
apnsAlias = "key alias"
apnsPass = "key password"
connect(user, password, url)
svc = getOpssService(name='KeyStoreService')
svc.importKeyStore(appStripe='system', name='RetailAppsPushServices', type='JKS',
filepath=keypath, password=keypass, permission=true, aliases=apnsAlias,
keypasswords=apnsPass)
exit('y')
```

For more details on how to import the created java keystores into weblogic, refer to Oracle WebLogic Server and Oracle Fusion Middleware documentation.

For Android notifications, retailers will need to set up the acquired API key as a generic credential in WebLogic. The GCM API key must be set in a map named "RetailAppsPushServices" under the key "gcmApiKey". Refer to Oracle WebLogic Server and Oracle Fusion Middleware documentation for instructions on creating and setting generic credentials

Lastly, if the server environment has proxy settings, the RetailAppsPushServices deployed on the weblogic server has to be modified. Retrieve the RetailAppsPushServices application enterprise archive (EAR) from the server, and modify the proxy.properties within. Once completed, the modified EAR file has to be redployed to the server.

Managing Application Navigator

Retail Applications provide an ability to switch between applications using the Application Navigator facility. These applications are configured using the Manage Application Navigator screens on Retail Application Administration Console (RAAC). For more details on Application Navigator in RAAC, please refer to the *Oracle Retail Merchandising Implementation Guide*.

Managing Functional Security

This section discusses the functional security for Retail applications and the components used to implement it. Functional security is based on OPSS. For more information on OPSS, refer to the *Oracle Fusion Middleware Application Security Guide*.

This section covers the following topics:

Introduction to Retail Roles

Users are not assigned to permissions directly; rather access is assigned to roles. Roles group particular permissions required to accomplish a task; instead of assigning individual permissions, roles match users with the permissions required to complete their particular task.

There are two main types of roles, enterprise and application.

The Identity Store contains enterprise roles that are available across applications. These are created as groups in LDAP, making them available across applications.

Applicable Retail Applications security provides four types of roles: abstract, job, duty, and privilege.

Applicable Retail Applications will record job, abstract roles as enterprise roles and duty, privilege roles as application roles.

Security Policy Stripe

Application roles are stored in the application-specific policy store. These roles and role mappings are described in the jazn-data.xml file under the policy stripe [Reim].

Abstract Roles

Abstract roles are associated with a user, irrespective of their job or job function. These roles are not associated with a job or duty. These roles are normally assigned by the system (based on user attributes), but can be provisioned to a user on request.

Naming Convention: All the Retail Abstract role names end with' _ABSTRACT'

Example: APPLICATION_ADMIN_ABSTRACT

Job Roles

Job roles are associated with the job of a user. A user with this job can have many job functions or job duties.

> **Note:** These roles are called Job roles as the role names closely map to the jobs commonly found in most organizations.

Naming Convention: All the Retail Job role names end with' _JOB'

Example: REIM_APPLICATION_ADMINISTRATOR_JOB.

Duty Roles

Job duties are tasks one must do on a job. A person is hired into a job role. These are the responsibilities one has for a job.

Duty roles are roles that are associated with a specific duty or a logical grouping of tasks. Generally, the list of duties for a job is a good indicator of what duty roles should be defined.

Duty roles should:

- Read as a job description at a job posting site.
- Duties that we create should be self-contained and pluggable into any existing or new job or abstract role.

Naming Convention: All the Retail duty role names end with' _DUTY'

Example: SUPPLIER_OPTIONS_MAINTENANCE_DUTY

Privilege Roles

Privilege is the logical collection of permissions. A privilege can be associated with any number of User Interface components. Privileges are expressed as application roles.

Naming Convention: All the Retail Privilege role names end with' _PRIV'

Example: SEARCH_SUPPLIER_OPTIONS_PRIV

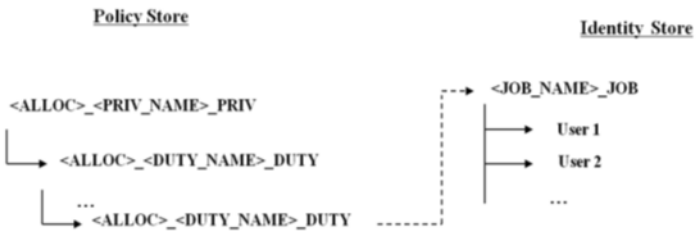
```
Privilege roles carry security grants.
Example:
<grant>
   <grantee>
      <principals>
            <principal>
<class>oracle.security.jps.service.policystore.
ApplicationRole</class>
               <name>MAINTAIN_DOCUMENTS_PRIV</name>
            </principal>
       </principals>
     </grantee>
  <permissions>
   <permission>

<class>oracle.adf.controller.security.TaskFlowPermission</class>

<name>/oracle/retail/apps/reim/view/documentmaintenance/flow/DocumentMaintenanceFl
```

```
ow.xml#DocumentMaintenanceFlow</name>
                                    <actions>view</actions>
                                </permission>
 </permissions>
</grant>
```

Retail Role Hierarchy

Retail role hierarchies are structured to reflect the Retail business process model.

Figure 3–3 Retail Role Hierarchy



Job roles inherit duty roles. For example, the ReIM application administrator Job role inherits the SYSTEM_OPTIONS_MAINTENANCE_DUTY roles.

```
<app-role>
  <name> SYSTEM_OPTIONS_MAINTENANCE_DUTY </name>
  <class>oracle.security.jps.service.policystore.ApplicationRole</class>
  <members>
   <member>
     <class>oracle.security.jps.internal.core.principals.
               JpsXmlEnterpriseRoleImpl</class>
     <name> REIM_APPLICATION_ADMINISTRATOR_JOB </name>
  </member>
 </members>
</app-role>
```

Duty roles inherit Privilege roles. Duty roles can inherit one or more other Duty roles.

Example: DOCUMENT_MANAGEMENT_DUTY inherits DOCUMENT_INQUIRY_ DUTY role.

```
<app-role>
     <name>DOCUMENT_INQUIRY_DUTY</name>
     <class>oracle.security.jps.service.policystore.ApplicationRole</class>
     <members>
         <member>

<class>oracle.security.jps.service.policystore.ApplicationRole</class>
             <name>DOCUMENT_MANAGEMENT_DUTY</name>
         </member>
         <member>

<class>oracle.security.jps.internal.core.principals.JpsXmlEnterpriseRoleImpl</clas
s>
             <name>REIM_APPLICATION_ADMINISTRATOR_JOB</name>
         </member>
     </members>
```

```
 </app-role>
```

Example: DOCUMENT_INQUIRY_DUTY role inherits the SEARCH_DOCUMENTS_PRIV role

```
<app-role>
        <name>SEARCH_DOCUMENTS_PRIV</name>
        <class>oracle.security.jps.service.policystore.ApplicationRole</class>
        <members>
            <member>

<class>oracle.security.jps.service.policystore.ApplicationRole</class>
                <name>DOCUMENT_INQUIRY_DUTY</name>
            </member>
            <member>

<class>oracle.security.jps.internal.core.principals.JpsXmlEnterpriseRoleImpl</clas
s>
                <name>REIM_APPLICATION_ADMINISTRATOR_JOB</name>
            </member>
        </members>
   </app-role>
```

Default Security Reference Implementation

Retail applications ship with default security reference implementations. The source of truth for default reference implementation is the jazn-data.xml file.

Privileges

Table 3–3 Privileges

Name	Description
Search Documents Priv	A privilege for searching for documents
Maintain Documents Priv	A privilege for creating and editing documents.
Delete Documents Priv	A privilege for deleting documents.
View Documents Priv	A privilege for viewing documents,
Pay Invoice Manually Priv	A privilege to allow user to pay an invoice before matching the invoice.
EDI Maintenance Priv	A privilege for allowing a user to fix EDI errors.
Reverse Debit Memo Priv	A privilege for allowing a user to reverse a Debit Memo
Create Credit Note from CNR Priv	A privilege for allowing a user to create a credit note from a credit note request
Void Credit Note Priv	A privilege for allowing a user to void a credit note.
Unmatch Priv	A privilege for allowing a user to unmatch a merchandise invoice.
Upload_Documents_Priv	A privilege for allowing a user to upload documents.
Search Financal Posting Error Priv	A privilege for searching for financial posting errors.
Maintain Financial Posting Error Priv	A privilege for editing financial posting errors.

Table 3–3 (Cont.) Privileges

Name	Description
View Financial Posting Error Priv	A privilege for viewing financial posting errors. .
Search Tolerance Priv	A privilege for searching for tolerance settings.
Maintain Tolerance Priv	A privilege for creating and editing tolerance settings.
Delete Tolerance Priv	A privilege for deleting a tolerance setting
View Tolerance Priv	A privilege for viewing for tolerance settings.
Search Tolerance Mapping Priv	A privilege for searching for tolerance mapping settings.
Maintain Tolerance Mapping Priv	A privilege for creating and editing tolerance mapping settings.
Delete Tolerance Mapping Priv	A privilege for deleting a tolerance mapping setting
View Tolerance Mapping Priv	A privilege for viewing for tolerance mapping settings.
Search Match Strategy Priv	A privilege for searching for Match Strategy settings.
Maintain Match Strategy Priv	A privilege for creating and editing Match Strategy settings.
Delete Match Strategy Priv	A privilege for deleting a Match Strategy setting
View Match Strategy Priv	A privilege for viewing for Match Strategy settings.
Search Match Strategy Mapping Priv	A privilege for searching for Match Strategy mapping settings.
Maintain Match Strategy Mapping Priv	A privilege for creating and editing Match Strategy mapping settings.
Delete Match Strategy Mapping Priv	A privilege for deleting a Match Strategy mapping setting
View Match Strategy Mapping Priv	A privilege for viewing for Match Strategy mapping settings.
Search G/L Cross Reference Priv	A privilege for searching for G/L Cross Reference settings.
Maintain G/L Cross Reference Priv	A privilege for creating and editing t G/L Cross Reference settings.
Delete G/L Cross Reference Priv	A privilege for deleting a G/L Cross Reference setting
View G/L Cross Reference Priv	A privilege for viewing for G/L Cross Reference settings.
Search Reason Code Priv	A privilege for searching for Reason Code settings.
Maintain Reason Code Priv	A privilege for creating and editing Reason Code settings.
Delete Reason Code Priv	A privilege for deleting a Reason Code setting
View Reason Code Priv	A privilege for viewing for Reason Code settings.
Search G/L Options Priv	A privilege for searching for G/L Options settings.
Maintain G/L Options Priv	A privilege for creating and editing G/L Options settings.
Delete G/L Options Priv	A privilege for deleting a G/L Options setting
View G/L Options Priv	A privilege for viewing for G/L Options settings.

Table 3–3 (Cont.) Privileges

Name	Description
Maintain System Options Priv	A privilege for creating and editing System Options settings.
Delete System Options Priv	A privilege for deleting a System Options setting
View System Options Priv	A privilege for viewing for System Options settings.
Search Supplier Options Priv	A privilege for searching for Supplier Options settings.
Maintain Supplier Options Priv	A privilege for creating and editing Supplier Options settings.
Delete Supplier Options Priv	A privilege for deleting a Supplier Options setting
View Supplier Options Priv	A privilege for viewing for Supplier Options settings.
Search Location Dynamic Mapping Priv	A privilege for searching for Location Dynamic Mapping settings.
Maintain Location Dynamic Mapping Priv	A privilege for creating and editing Location Dynamic Mapping settings.
Delete Location Dynamic Mapping Priv	A privilege for deleting a Location Dynamic Mapping setting
View Location Dynamic Mapping Priv	A privilege for viewing for Location Dynamic Mapping settings.
Search Dept Class Dynamic Mapping Priv	A privilege for searching for Dept Class Dynamic Mapping settings.
Maintain Dept Class Dynamic Mapping Priv	A privilege for creating and editing Dept Class Dynamic Mapping settings.
Delete Dept Class Dynamic Mapping Priv	A privilege for deleting a Dept Class Dynamic Mapping setting
View Dept Class Dynamic Mapping Priv	A privilege for viewing for Dept Class Dynamic Mapping settings.
Search Manual Match Priv	A privilege to allow a user search for invoices and receipts to be manually matched
Manual Match Priv	A privilege to allow a user to match an invoice manually through the UI.
Search Credit Note Match Priv	A privilege to allow a user to search for Credit Notes to Credit Note Requests to be matched,.
Credit Note Match Priv	A privilege to match Credit Notes to Credit Note Requests.
Search Discrepancy List Priv	A privilege to search for Discrepancies.
Cost Resolution Priv	A privilege to allow a cost discrepancy to be resolved.
Quantity Resolution Priv	A privilege to allow a quantity discrepancy to be resolved
Search Tax Discrepancy List Priv	A privilege to search for Tax Discrepancies
Tax Discrepancy Resolution Priv	A privilege to allow a tax discrepancy to be resolved.

Duties

Table 3–4 Duties

Name	Description	List of Privileges
Document Management Duty	A Duty for managing documents. This duty is an extension of the Document Inquiry Duty	▪ All privileges found in the Document Inquiry Duty. ▪ Maintain Documents Privilege ▪ Delete Documents Privilege
Document Inquiry Duty	A duty for viewing documents	-View Documents Priv -Search Documents Priv
Pay Invoice Duty	A duty for paying an invoice manually	-Pay Invoice Manually Priv
EDI Maintenance Duty	A duty for fixing EDI Errors.	-EDI Maintenance Priv
Reverse Debit Memo Duty	A duty to allow user to reverse a Credit Note	- Reverse Debit Memo Priv
Create Credit Note from CNR Duty	A duty to allow user to create a Credit Note from a Credit Note Request	- Create Credit Note from CNR Priv
Void Credit Note Duty	A duty for voiding a Credit Note.	- Void Credit Note Priv
Unmatch Duty	A duty for unmatching a merchandise invoice	Unmatch Priv
Upload Duty	A duty for uploading documents	Unload Documents Priv
Financial Posting Error Duty	A Duty for managing Financial Posting Errors. This duty is an extension of the Financial Posting Errors Inquiry Duty	-All privileges found in the Financial Posting Error Inquiry Duty. -Maintain Financial Posting Error Priv
Financial Posting Error Inquiry Duty	A duty for viewing Financial Posting Errors	-View Financial Posting Error Priv -Search Financial Posting Error Priv

Table 3–4 (Cont.) Duties

Name	Description	List of Privileges
Tolerance Maintenance Duty	A Duty for managing Tolerance settings. This duty is an extension of the Tolerance Inquiry Duty	-All privileges found in the Tolerance Inquiry Duty. -Maintain Tolerance Priv -Delete Tolerance Priv -Maintain Tolerance Mapping Priv -Delete Tolerance Mapping Priv
Tolerance Inquiry Duty	A duty for viewing Tolerance settings	-View Tolerance Priv -View Tolerance Mapping Priv -Search Tolerance Priv -Search Tolerance Mapping Priv
Match Strategy Maintenance Duty	A Duty for managing Match Strategy settings. This duty is an extension of the Match Strategy Inquiry Duty	-All privileges found in the Match Strategy Inquiry Duty. -Maintain Match Strategy Priv -Delete Match Strategy Priv -Maintain Match Strategy Mapping Priv -Delete Match Strategy Mapping Priv
Match Strategy Inquiry Duty	A duty for viewing Match Strategy settings	-View Match Strategy Priv -View Match Strategy Mapping Priv -Search Match Strategy Priv -Search Match Strategy Mapping Priv
G/L Cross Reference Maintenance Duty	A Duty for managing G/L Cross Reference settings. This duty is an extension of the G/L Cross Reference Inquiry Duty	-All privileges found in the G/L Cross Reference Inquiry Duty. -Maintain G/L Cross Reference Priv -Delete G/L Cross Reference Priv
G/L Cross Reference Inquiry Duty	A duty for viewing G/L Cross Reference settings	-View G/L Cross Reference Priv -Search G/L Cross Reference Priv
Reason Code Maintenance Duty	A Duty for managing Reason Code settings. This duty is an extension of the Reason Code Inquiry Duty	-All privileges found in the Reason Code Inquiry Duty. -Maintain Reason Code Priv -Delete Reason Code Priv
Reason Code Inquiry Duty	A duty for viewing Reason Code settings	-View Reason Code Priv -Search Reason Code Priv

Table 3–4 (Cont.) Duties

Name	Description	List of Privileges
G/L Options Maintenance Duty	A Duty for managing G/L Options settings. This duty is an extension of the G/L Options Inquiry Duty	-All privileges found in the G/L Options Inquiry Duty. -Maintain G/L Options Priv -Delete G/L Options Priv
G/L Options Inquiry Duty	A duty for viewing G/L Options settings	-View G/L Options Priv -Search G/L Options Priv
System Options Maintenance Duty	A Duty for managing System Options settings. This duty is an extension of the System Options Inquiry Duty	-All privileges found in the System Options Inquiry Duty. -Maintain System Options Priv -Delete System Options Priv
System Options Inquiry Duty	A duty for viewing System Options settings	-View System Options Priv
Supplier Options Maintenance Duty	A Duty for managing Supplier Options settings. This duty is an extension of the Supplier Options Inquiry Duty	-All privileges found in the Supplier Options Inquiry Duty. -Maintain Supplier Options Priv -Delete Supplier Options Priv
Supplier Options Inquiry Duty	A duty for viewing Supplier Options settings	-View Supplier Options Priv -Search Supplier Options Priv
Location Dynamic Mapping Maintenance Duty	A Duty for managing Location Dynamic Mapping settings. This duty is an extension of the Location Dynamic Mapping Inquiry Duty	-All privileges found in the Location Dynamic Mapping Inquiry Duty. -Maintain Location Dynamic Mapping Priv -Delete Location Dynamic Mapping Priv
Location Dynamic Mapping Inquiry Duty	A duty for viewing Location Dynamic Mapping settings	-View Location Dynamic Mapping Priv -Search Location Dynamic Mapping Priv

Table 3–4 (Cont.) Duties

Name	Description	List of Privileges
Dept Class Dynamic Mapping Maintenance Duty	A Duty for managing Dept Class Dynamic Mapping settings. This duty is an extension of the Dept Class Dynamic Mapping Inquiry Duty	-All privileges found in the Dept Class Dynamic Mapping Inquiry Duty. -Maintain Dept Class Dynamic Mapping Priv -Delete Dept Class Dynamic Mapping Priv
Dept Class Dynamic Mapping Inquiry Duty	A duty for viewing Dept Class Dynamic Mapping settings	-View Dept Class Dynamic Mapping Priv -Search Dept Class Dynamic Mapping Priv
Invoice Matching Duty	A duty for manually matching invoices	-Manually Match Priv -Search Manual Match Priv
Resolve Cost Discrepancies Duty	A duty for resolving cost discrepancies	-Cost Discrepancy Priv -Search Discrepancy List Priv
Resolve Quantity Discrepancies Duty	A duty for resolving quantity discrepancies	-Quantity Discrepancy Priv -Search Discrepancy List Priv
Resolve Discrepancies Duty	A duty for resolving either cost or quantity discrepancies	-Cost Discrepancy Priv -Quantity Discrepancy Priv -Search Discrepancy List Priv
Credit Note Matching Duty	A duty for matching Credit Notes	-Credit Note Match Priv -Search Credit Note Match Priv-
Resolve Tax Discrepancy Duty	A duty for resolving tax discrepancies	-Tax Discrepancy Resolution Priv -Search Tax Discrepancy List Priv-

Role Mapping

Table 3–5 Role Mapping

Role	Duty	Privileges
Accounts Payable Specialist	Document Management Duty	-All privileges found in the Document Inquiry Duty.
	Pay Invoice Duty	-Maintain Documents Priv
	EDI Maintenance Duty	-Delete Documents Priv
	Reverse Debit Memo Duty	-Pay Invoice Manually Priv
	Create Credit Note from CNR Duty	-EDI Maintenance Priv
	Void Credit Note Duty	-Reverse Debit Memo Priv
	Tolerance Inquiry Duty	-Create Credit Note from CNR Priv
	Match Strategy Inquiry Duty	-Void Credit Note Priv
	G/L Cross Reference Inquiry Duty	-View Tolerance Priv
	Reason Code Inquiry Duty	-View Tolerance Mapping Priv
	G/L Options Inquiry Duty	-Search Tolerance Priv
	System Options Inquiry Duty	-Search Tolerance Mapping Priv
	Supplier Options Inquiry Duty	-View Match Strategy Priv
	Location Dynamic Mapping Inquiry Duty	-View Match Strategy Mapping Priv
	Dept Class Dynamic Mapping Inquiry Duty	-Search Match Strategy Priv
	Invoice Matching Duty	-Search Match Strategy Mapping Priv
	Credit Note Matching Duty	-View G/L Cross Reference Priv
	Discrepancy Resolution Duty	-Search G/L Cross Reference Priv
	Override Department Security Duty	-View Reason Code Priv
	Resolve Tax Discrepancy Duty	-Search Reason Code Priv
		-View G/L Options Priv
		-Search G/L OptionsPriv
		-View System Options Priv
		-Search System Options Priv
		-View Supplier Options Priv
		-Search Supplier Options Priv
		-View Location Dynamic Mapping Priv
		-Search Location Dynamic Mapping Priv
		-View Dept Class Dynamic Mapping Priv
		-Search Dept Class Dynamic Mapping Priv
		-Manually Matching Priv
		-Cost Discrepancy Priv
		-Quantity Discrepancy Priv
		-Override Department Security Priv
		-Credit Note Matching Priv-

Table 3–5 (Cont.) Role Mapping

Role	Duty	Privileges
Finance Manager	Document Management Duty	-All privileges found in the Document Inquiry Duty.
		-Maintain Documents Priv
	Pay Invoice Duty	-Delete Documents Priv
	EDI Maintenance Duty	-Pay Invoice Manually Priv
		-EDI Maintenance Priv
	Reverse Debit Memo Duty	-Reverse Debit Memo Priv
	Create Credit Note from CNR Duty	-Create Credit Note from CNR Priv
		-Void Credit Note Priv
	Void Credit Note Duty	-All privileges found in the Tolerance Inquiry Duty.
		-Maintain Tolerance Priv
	Tolerance Maintenance Duty	-Delete Tolerance Priv
	Match Strategy Maintenance Duty	-Maintain Tolerance Mapping Priv
		-Delete Tolerance Mapping Priv
	G/L Cross Reference Maintenance Duty	-All privileges found in the Match Strategy Inquiry Duty.
		-Maintain Match Strategy Priv
	Reason Code Maintenance Duty	-Delete Match Strategy Priv
		-Maintain Match Strategy Mapping Priv
	G/L Options Maintenance Duty	-Delete Match Strategy Mapping Priv
	System Options Inquiry Duty	-All privileges found in the G/L Cross Reference Inquiry Duty.
		-Maintain G/L Cross Reference Priv
	Supplier Options Maintenance Duty	-Delete G/L Cross Reference Priv
	Location Dynamic Mapping Maintenance Duty	-All privileges found in the Reason Code Inquiry Duty.
		-Maintain Reason Code Priv
		-Delete Reason Code Priv
	Dept Class Dynamic Mapping Maintenance Duty	-All privileges found in the G/L Options Inquiry Duty.
		-Maintain G/L Options Priv
	Invoice Matching Duty	-Delete G/L Options Priv
	Credit Note Matching Duty	-All privileges found in the System Options Inquiry Duty.
	Discrepancy Resolution Duty	-All privileges found in the Supplier Options Inquiry Duty.
		-Maintain Supplier Options Priv
	Override Department Security Duty	-Delete Supplier Options Priv
	Resolve Tax Discrepancy Duty	-All privileges found in the Location Dynamic Mapping Inquiry Duty.
		-Maintain Location Dynamic Mapping Priv
		-Delete Location Dynamic Mapping Priv
		-All privileges found in the Dept Class Dynamic Mapping Inquiry Duty.
		-Maintain Dept Class Dynamic Mapping Priv
		-Delete Dept Class Dynamic Mapping Priv

Table 3–5 (Cont.) Role Mapping

Role	Duty	Privileges
		-Manually Match Priv
		-Search Manual Match Priv
		-Credit Note Match Priv
		-Search Credit Note Match Priv
		-Cost Discrepancy Priv
		-Quantity Discrepancy Priv
		-Search Discrepancy List Priv
		-Override Department Security Priv
		-Tax Discrepancy Resolution Priv
		-Search Tax Discrepancy List Priv
Buyer	Override Department Security Duty	-Cost Discrepancy Priv
		-Search Discrepancy List Priv
	Resolve Cost Discrepancies Duty	-Override Department Security Priv
Corporate Inventory Control Analyst	Resolve Quantity Discrepancies Duty	-Quantity Discrepancy Priv
		-Search Discrepancy List Priv
ReIM Application Administrator	All duties.	-All privileges
Financial Analyst	Document Inquiry Duty	-All privileges in
		-Document Inquiry Duty.
	Pay Invoice Duty	-All privileges in
	Tolerance Inquiry Duty	-Tolerance Inquiry Duty.
	Match Strategy Inquiry Duty	-All privileges found in the Match Strategy Inquiry Duty.
	G/L Cross Reference Inquiry Duty	-All privileges found in the G/L Cross Reference Inquiry Duty.
	Reason Code Inquiry Duty	-All privileges found in the Reason Code Inquiry Duty.
	G/L Options Inquiry Duty	-All privileges found in the G/L Options Inquiry Duty.
	System Options Inquiry Duty	-All privileges found in the System Options Inquiry Duty.
	Supplier Options Inquiry Duty	-All privileges found in the Supplier Options Inquiry Duty.
	Location Dynamic Mapping Inquiry Duty	-All privileges found in the Location Dynamic Mapping Inquiry Duty.
	Dept Class Dynamic Mapping Inquiry Duty	-All privileges found in the Dept Class Dynamic Mapping Inquiry Duty.
	Financial Posting Error Inquiry Duty	-All privileges found In the Financial Posting Error Inquiry Duty
		-

Table 3–5 (Cont.) Role Mapping

Role	Duty	Privileges
Accounts Payable Manager	Document Management Duty	-EDI Maintenance Priv
	Pay Invoice Duty	-Reverse Debit Memo Priv
	EDI Maintenance Duty	-Create Credit Note from CNR Priv
	Reverse Debit Memo Duty	-Void Credit Note Priv
	Create Credit Note from CNR Duty	-All privileges found in the Tolerance Inquiry Duty.
	Void Credit Note Duty	-Maintain Tolerance Priv
	Tolerance Maintenance Duty	-Delete Tolerance Priv
	Match Strategy Maintenance Duty	-Maintain Tolerance Mapping Priv
	G/L Cross Reference Maintenance Duty	-Delete Tolerance Mapping Priv
	Reason Code Maintenance Duty	-All privileges found in the Match Strategy Inquiry Duty.
	G/L Options Maintenance Duty	-Maintain Match Strategy Priv
	System Options Inquiry Duty	-Delete Match Strategy Priv
	Supplier Options Maintenance Duty	-Maintain Match Strategy Mapping Priv
	Location Dynamic Mapping Maintenance Duty	-Delete Match Strategy Mapping Priv
	Dept Class Dynamic Mapping Maintenance Duty	-All privileges found in the G/L Cross Reference Inquiry Duty.
	Invoice Matching Duty	-Maintain G/L Cross Reference Priv
	Credit Note Matching Duty	-Delete G/L Cross Reference Priv
	Discrepancy Resolution Duty	-All privileges found in the Reason Code Inquiry Duty.
	Resolve Tax Discrepancy Duty	-Maintain Reason Code Priv
	Unmatch Duty	-Delete Reason Code Priv
	Upload Documents Duty	-All privileges found in the G/L Options Inquiry Duty.
	Financial Posting Error Duty	-Maintain G/L Options Priv
		-Delete G/L Options Priv
		-All privileges found in the System Options Inquiry Duty.
		-All privileges found in the Supplier Options Inquiry Duty.
		-Maintain Supplier Options Priv
		-Delete Supplier Options Priv
		-All privileges found in the Location Dynamic Mapping Inquiry Duty.
		-Maintain Location Dynamic Mapping Priv
		-Delete Location Dynamic Mapping Priv
		-All privileges found in the Dept Class Dynamic Mapping Inquiry Duty.
		-Maintain Dept Class Dynamic Mapping Priv
		-Delete Dept Class Dynamic Mapping Priv

Table 3–5 (Cont.) Role Mapping

Role	Duty	Privileges
		-Manually Match Priv
		-Search Manual Match Priv
		-Credit Note Match Priv
		-Search Credit Note Match Priv
		-Cost Discrepancy Priv
		-Quantity Discrepancy Priv
		-Search Discrepancy List Priv
		-Tax Discrepancy Resolution Priv
		-Search Tax Discrepancy List Priv
		-Unmatch Priv
		-Upload documents Priv
		-All privileges found In the Financial Posting Error Inquiry Duty
		-Financial Posting Error Priv

Table 3–5 (Cont.) Role Mapping

Role	Duty	Privileges
Data Steward	Document Management Inquiry Duty	-All privileges found in the Document Inquiry Duty.
	EDI Maintenance Duty	-EDI Maintenance Priv
	Tolerance Maintenance Duty	-All privileges found in the Tolerance Inquiry Duty.
	Match Strategy Maintenance Duty	-Maintain Tolerance Priv
	G/L Cross Reference Maintenance Duty	-Delete Tolerance Priv
	Reason Code Maintenance Duty	-Maintain Tolerance Mapping Priv
	G/L Options Maintenance Duty	-Delete Tolerance Mapping Priv
	System Options Maintenance Duty	-All privileges found in the Match Strategy Inquiry Duty.
	Supplier Options Maintenance Duty	-Maintain Match Strategy Priv
	Location Dynamic Mapping Maintenance Duty	-Delete Match Strategy Priv
	Dept Class Dynamic Mapping Maintenance Duty	-Maintain Match Strategy Mapping Priv
	Upload Documents Duty	-Delete Match Strategy Mapping Priv
	Financial Posting Error Duty	-All privileges found in the G/L Cross Reference Inquiry Duty.
		-Maintain G/L Cross Reference Priv
		-Delete G/L Cross Reference Priv
		-All privileges found in the Reason Code Inquiry Duty.
		-Maintain Reason Code Priv
		-Delete Reason Code Priv
		-All privileges found in the G/L Options Inquiry Duty.
		-Maintain G/L Options Priv
		-Delete G/L Options Priv
		-All privileges found in the System Options Inquiry Duty.
		-Maintain System Options Priv
		-All privileges found in the Supplier Options Inquiry Duty.
		-Maintain Supplier Options Priv
		-Delete Supplier Options Priv
		-All privileges found in the Location Dynamic Mapping Inquiry Duty.
		-Maintain Location Dynamic Mapping Priv
		-Delete Location Dynamic Mapping Priv
		-All privileges found in the Dept Class Dynamic Mapping Inquiry Duty.
		-Maintain Dept Class Dynamic Mapping Priv
		-Delete Dept Class Dynamic Mapping Priv
		-Upload documents Priv
		-All privileges found In the Financial Posting Error Inquiry Duty
		-Financial Posting Error Priv

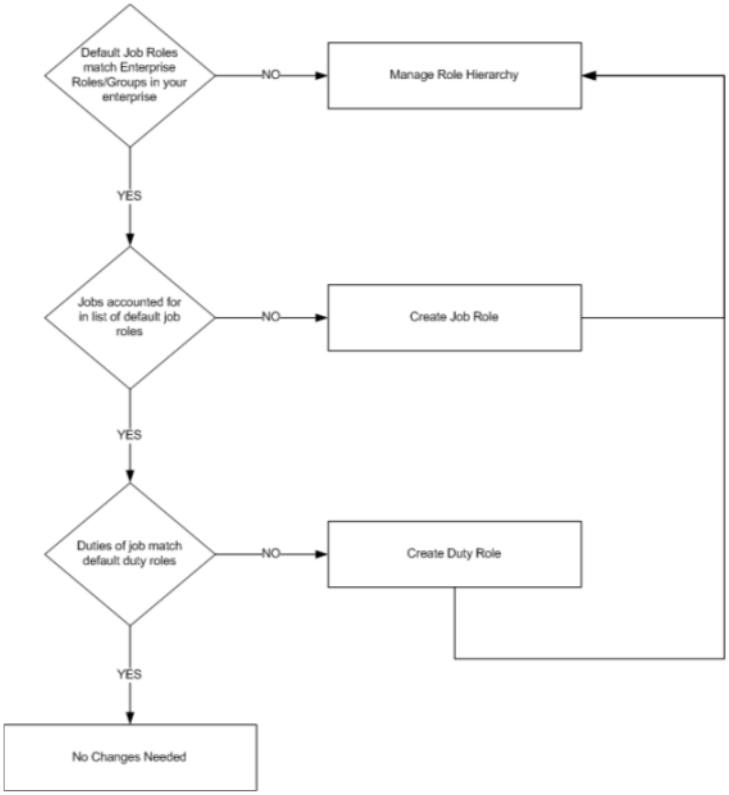
Extending the Default Security Reference Implementation

Note: Make sure that the policy store is loaded with the default security configuration. For more information, see the Post Installation steps in the *Oracle Retail Invoice Matching Installation Guide*.

The common decisions made to match your enterprise to the default security reference implementation include the following:

- Do the default job roles match the equivalent job roles in your enterprise?
- Do the jobs in your enterprise exist in the security reference implementation?
- Do the duties performed by the jobs in your enterprise match the duties in the security reference implementation?

Figure 3–4 Decisions for Default Security Reference Implementation



Important: It is important when constructing a role hierarchy that circular dependencies are not introduced. The best practice is to leave the default security configuration in place and first incorporate your customized application roles in a test environment.

Managing Roles in Retail Application Administration Console

Retail applications provide a way in which retailers can modify the default roles to map to their security groups through the Retail Application Administration Console (RAAC).

RAAC is installed along with the Retail Application. Users with proper security privileges to access RAAC can launch RAAC by clicking on a link from the Retail Application's global menu.

For more information about using RAAC, refer to the Oracle Retail Application Administration Console chapter of the *Oracle Retail Merchandising Implementation Guide*.

Disabling Content

There are situations where administrators need to disable certain links or the default content such as Dashboards due to unavailability or other reasons. Retail Applications provide the flexibility to disable such content so that the application remains largely unaffected.

Safe Mode

Applications can choose to serve certain content such as dashboards to users upon launching the application. This is referred to as "Default Content". However sometimes this default content may cause delays in application launch after logging-in or worse it may render the application unusable.

To handle such scenarios Retail Applications provide a feature for Administrators called "Safe Mode" which. allows the user to log in without serving up any default content. Once this mode is turned on, no default content is shown to any user when the application is launched.

To turn on this mode the property "uishell.load.safe.mode" must be set to true in the RetailAppsViewController.properties file.

Disabling Links in the Sidebar

Administrators may occasionally need to disable content launchable from links in the sidebar menu. Retail Applications provide the ability to disable such links.

To disable a link the Administrator must first find the "id" of that link as specified in the SidebarNavigationModel.xml file. This value must then be provided to the property "uishell.sidebar.invalid.item.ids" within the RetailAppsViewController.properties file. To disable more than one link, pass in multiple ids separated by a comma.

Managing Oracle Metadata Services (MDS)

Retail Applications are built using ADF and one of the features within ADF is the Oracle Metadata Services (MDS) framework which provides a facility for retailers to customize the applications.

Refer to the document, *Oracle® Fusion Middleware Developing Fusion Web Applications with Oracle Application Development Framework* (https://docs.oracle.com/middleware/1221/adf/develop/GUID-33CB77F6-6ECD-4B5D-9B95-5874AEDA1A78.htm#ADFFD2077), for more information about MDS.

Overview of Oracle Metadata Services

Oracle Metadata Services (MDS) is a key infrastructure component in Oracle Fusion Middleware. It is the layer through which metadata is loaded, saved, cached, stored, managed, and customized both by various middleware components and by the applications built on Fusion Middleware.

The use of MDS in ADF applications, for example, can allow applications to remember how users like to work, and therefore not require them to set up the application for every session. This may include, for example, saving of common searches and screen layouts for every user. This allows making use of the application easier and more intuitive for the users. MDS provides a foundation that can be leveraged by Oracle Application Development Framework (ADF) applications to provide such persistent personalization.

Oracle Metadata Services (MDS) makes use of metadata repositories or partitions. A Metadata repository or partition contains metadata for Oracle Fusion Middleware components. It can also contain metadata about the configuration of Oracle Fusion Middleware and metadata for applications. Oracle Metadata Services (MDS) stores the customizations in a metadata repository and retrieves them at runtime to merge the customizations with the base metadata to reveal the customized application.

A common problem when a patch is installed for a Retail Application is that certain screens would fail to load or UI elements fail to display data properly.

The cause of this issue is commonly attributed to user personalization on screen elements that are now removed in the patch.

For example, prior to patching the application, users may have saved search criterias on certain screens as a way to conveniently recall their desired search results whenever they use the application. Those saved search criterias are persisted by ADF in the MDS repository. If the patch involves the removal of one of the attributes used in the search criterias, applying the patch will cause the screens that have those search criterias fail to load.

The MDS repository is configured in the WebLogic server where the Retail Application is deployed. The repository is database-based and it is organized or subdivided into partitions. Retail Applications are deployed with their own partition within the server's MDS repository.

It is recommended to not delete the MDS partition during the upgrade of the Retail Application, instead use the functions described in this document to resolve any issues related to MDS.

Using the System MBean Browser and the MDSAppRuntime MBean

For managing MDS Customizations in Retail Fusion Applications, use the Oracle Enterprise Manager to perform common metadata service tasks such as exporting, deleting, and importing of MDS Customizations. This can be done in the Oracle Enterprise Manager using the System MBean Browser and the MDSAppRuntime MBean.

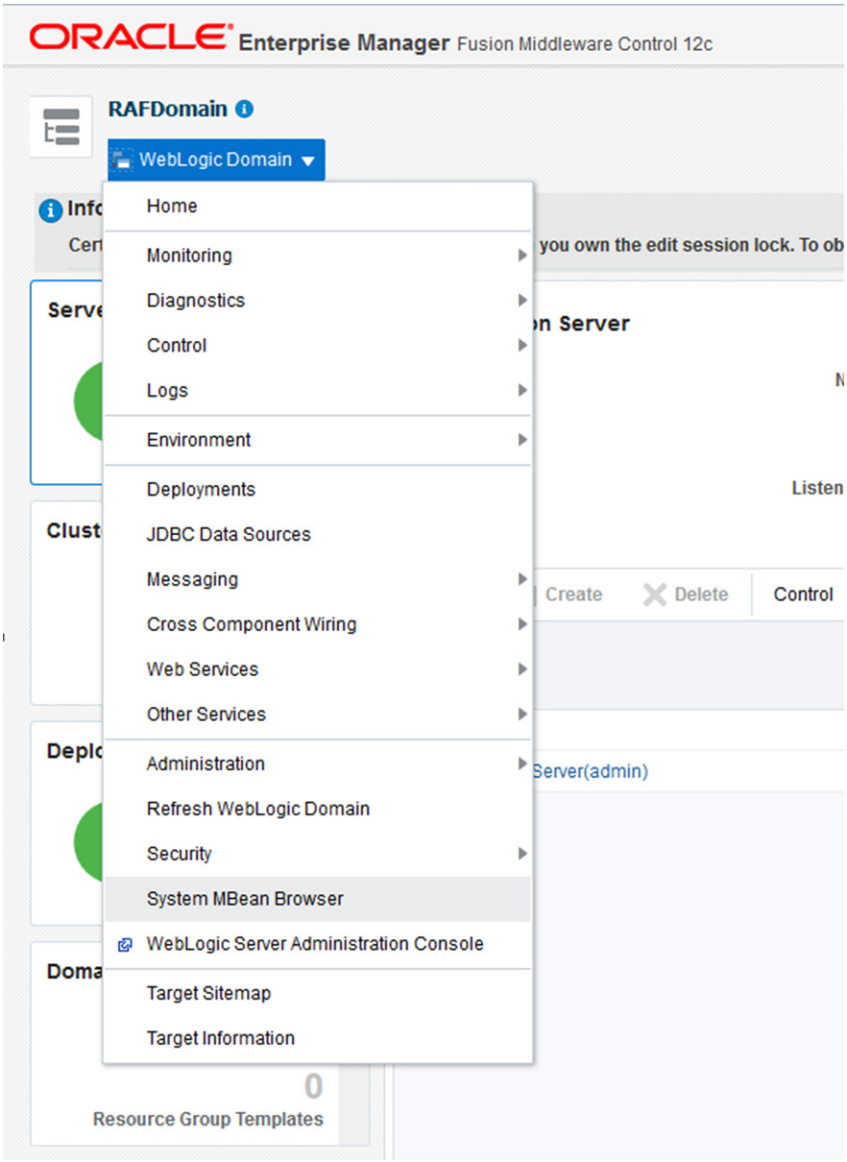
Perform the following steps to access the MDSAppRuntime MBean.

1. Login to the Oracle Enterprise Manager by navigating to the URL in the following format:

   http://<host>:<port>/em/

2. Click on the WebLogic Domain dropdown under the domain name. Navigate to System MBean Browser menu item and click on System MBean Browser. The screenshot below shows the RAFDomain as the domain name.

Figure 3–5 RAFDomain window

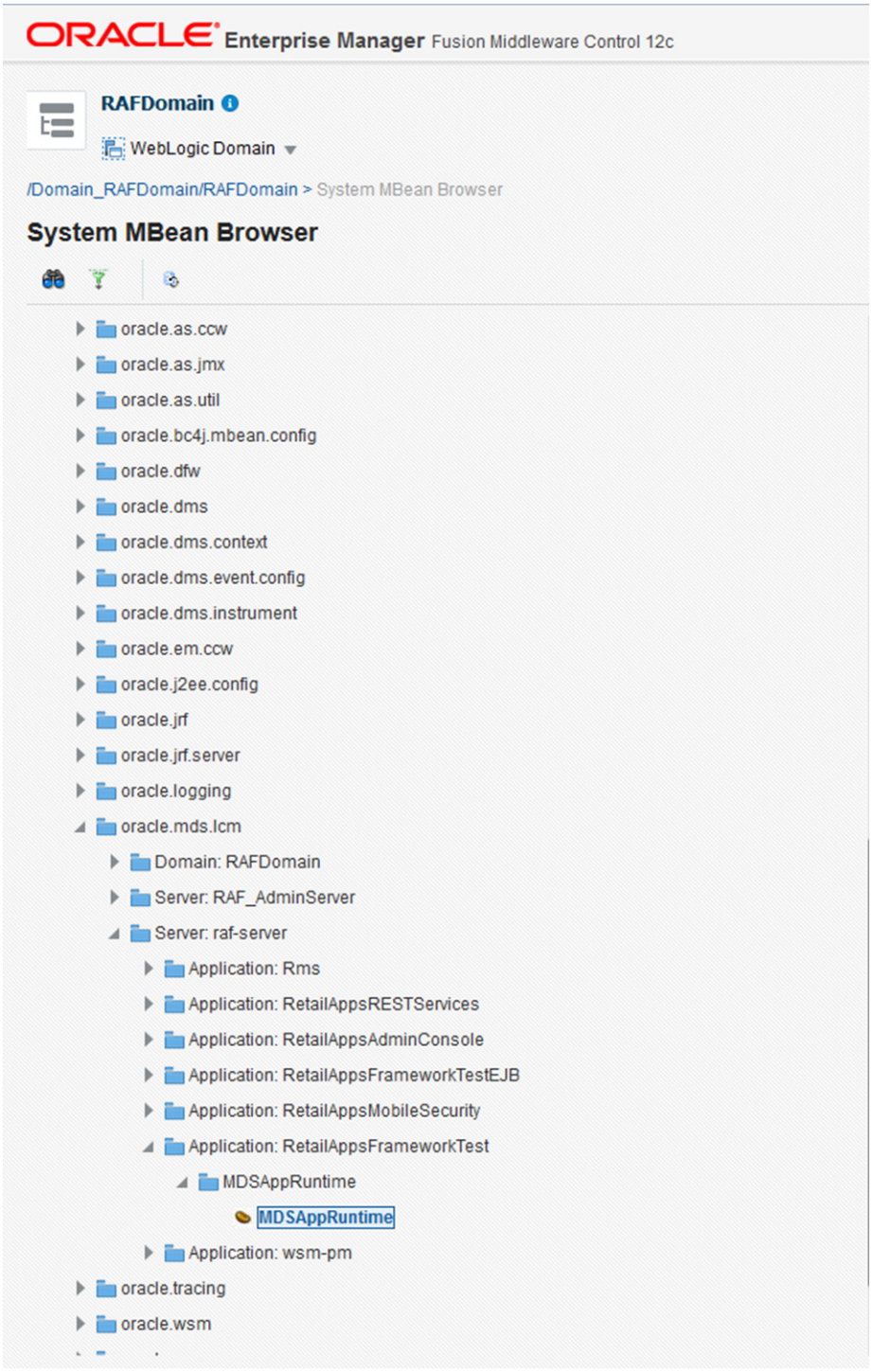


3. Under Application Defined MBeans, locate MDSAppRuntime which can be found under the following folder structure:

```
> oracle.mds.lcm
  > Server: AppServer
    > Application: Application Name
      > MDSAppRuntime
```

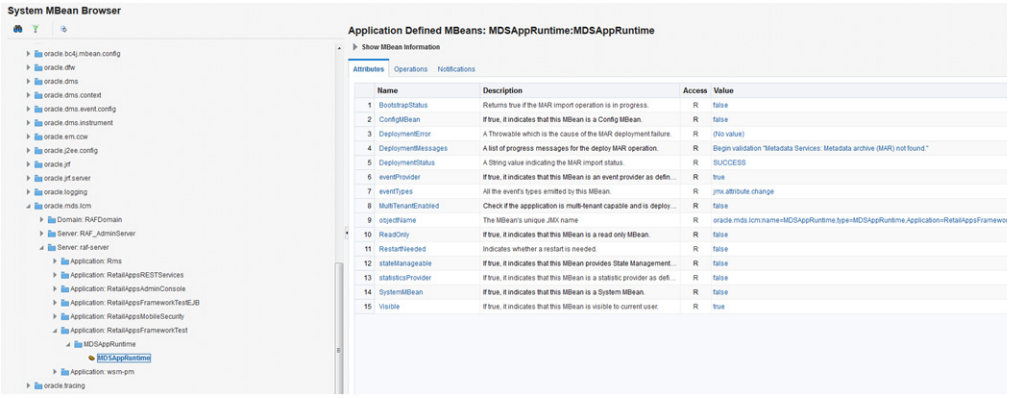
Choose the correct server and the Retail Application name based on your installation. The screenshot below displays RAFServer and RetailAppsFrameworkTest application.

Figure 3–6 RAFDomain System MBean Browser window



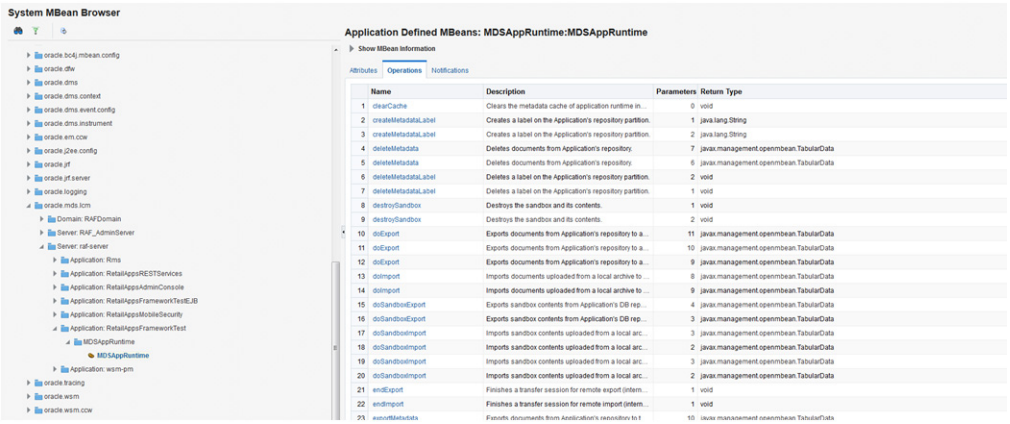
4. Click MDSAppRuntime management bean (MBean).

Figure 3–7 System MBean Browser



5. Select the Operations tab to view the operations available for the MDSAppRuntime MBean.

Figure 3–8 MBean Browser Operations Tab



These are the operations which can be used in managing MDS Customizations. The exportMetadata, deleteMetadata, importMetadata, createMetadataLabel, listMetadataLabels, deleteMetadataLabel, and promoteMetadataLabel; will be briefly discussed in the following sections.

Exporting All Metadata Services Customizations

1. Navigate to the MDSAppRuntime management bean (MBean), as described in the section Using the System MBean Browser and the MDSAppRuntime MBean.
2. Export metadata by selecting the exportMetadata operation available from the Operations tab.
3. For toLocation, provide a valid absolute path to a directory or archive in the file system to which the selected documents will be exported. This location must be accessible from the machine where the application is running. If it does not exist, a directory will be created except that when the name ends with ".jar", ".JAR", ".zip", or ".ZIP", an archive will be created. Exporting metadata to an existing archive will overwrite the existing file.

   Example: /tempDir/downloads/mdsExport/RetailAppsFrameworkTestMDS.zip

4. Click **Invoke** (located at the upper-right corner of the page) to proceed with the export operation.
5. Click **Return** to return to the list of operations.

Exporting Metadata Services Customization for a Specific User

1. Navigate to the MDSAppRuntime management bean (MBean), as described in the section Using the System MBean Browser and the MDSAppRuntime MBean.
2. Export metadata by selecting the exportMetadata operation available from the Operations tab.
3. For toLocation, provide a valid absolute path to a directory or archive in the file system to which the selected documents will be exported. This location must be accessible from the machine where the application is running. If it does not exist, a directory will be created except that when the name ends with ".jar", ".JAR", ".zip", or ".ZIP", an archive will be created. Exporting metadata to an existing archive will overwrite the existing file.

   Example: /tempDir/downloads/mdsExport/RetailAppsFrameworkTestMDS.zip
4. For docs, click the pencil icon. On the Edit Parameter popup, click **Add**, and in the Element box, enter a list of comma-separated, fully qualified document names or document name patterns to export.

   To export customizations for a specific page, simply enter the fully qualified base document name in the Element box.

   Example:

   /oracle/retail/apps/framework/uishell/skin/page/TestTablesAndTrees.jsff

   You can provide the path to multiple documents to export, by clicking the Add button. Do not provide any docs in case you want to export all customizations for the user.
5. For restrictCustTo, click the pencil icon. On the Edit Parameter popup, click **Add**, and in the Element box, enter the list of customization layer names. This can be a list of comma-separated customization layer names used to restrict the operation to only customization documents that match the specified customization layers.

   Each customization layer name can contain, within a pair of brackets, optional customization layer values and value patterns separated by commas. Wildcards (%) may also be used for restricting the operation.

   For example:

Table 3–6 Customization Values

Value	Description
user[buyer]	Restricts operation to the user 'buyer'
user[buyer, betty_buyer]	Restricts operation to the users 'buyer' and 'betty_buyer'
user[bu%]	Restricts operation to user names that start with 'bu' (for example, buyer)
user[be%]	Restricts operation to user names that start with 'be' (for example, betty_buyer)
user[%bu%]	Restricts operation to users with 'bu' in the name (for example, buyer and betty_buyer)

6. Click **Invoke**, located at the upper-right corner of the page, to proceed with the export operation.
7. A confirmation message will display in the page. Click **Return** to return to the list of operations.

Deleting All Metadata Services Customizations for a User

1. Navigate to the MDSAppRuntime management bean (MBean), as described in the section Using the System MBean Browser and the MDSAppRuntime MBean.
2. Delete metadata by selecting the deleteMetadata operation available from the Operations tab.
3. For docs, click the pencil icon. On the Edit Parameter popup, click **Add**, and in the Element box, enter a list of comma-separated, fully qualified document names or document name patterns.

   To delete all customizations for a user, enter "/**" (without the quotes).

   This will recursively delete all customizations under "/" including any other customizations located in the folder(s) under it. Click **OK**.
4. For restrictCustTo, click the pencil icon. On the Edit Parameter popup, click **Add**, and in the Element box, enter the list of customization layer names. This can be used to restrict the operation to only customization documents that match the specified customization layers.

   Each customization layer name can contain, within a pair of brackets, optional customization layer values and value patterns separated by commas. Wildcards (%) may also be used for restricting the operation.

   For example:

Table 3–7 Customization Values

Value	Description
user[buyer]	Restricts operation to the user 'buyer'
user[buyer, betty_buyer]	Restricts operation to the users 'buyer' and 'betty_buyer'
user[bu%]	Restricts operation to user names that start with 'bu' (for example, buyer)
user[be%]	Restricts operation to user names that start with 'be' (for example, betty_buyer)
user[%bu%]	Restricts operation to users with 'bu' in the name (for example, buyer and betty_buyer)

5. Click **Invoke**, located at the upper-right corner of the page, to proceed with the delete operation.
6. A confirmation message will display in the page. Click **Return** to return to the list of operations.

Deleting a Customization for a Specific Page for All the Users

1. Navigate to the MDSAppRuntime management bean (MBean), as described in the section Using the System MBean Browser and the MDSAppRuntime MBean.
2. Delete metadata by selecting the deleteMetadata operation available from the Operations tab.

3. For excludeBaseDocs, select true. This is a Boolean value indicating whether to exclude base metadata documents from being deleted.
4. For docs, click the pencil icon. On the Edit Parameter popup, click **Add**, and in the Element box, enter a list of comma-separated, fully qualified document names or document name patterns.

   To delete all customizations for a user, enter "/**" (without the quotes).

   This will recursively delete all customizations under "/" including any other customizations located in the folder(s) under it. Click **OK**.
5. For restrictCustTo, click the pencil icon. On the Edit Parameter popup, click **Add**, and in the Element box, enter the list of customization layer names. This can be used to restrict the operation to only customization documents that match the specified customization layers.

   Each customization layer name can contain, within a pair of brackets, optional customization layer values and value patterns separated by commas. Wildcards (%) may also be used for restricting the operation.

   For example:

Table 3–8 Customization Values

Value	Description
user[buyer]	Restricts operation to the user 'buyer'
user[buyer, betty_buyer]	Restricts operation to the users 'buyer' and 'betty_buyer'
user[bu%]	Restricts operation to user names that start with 'bu' (for example, buyer)
user[be%]	Restricts operation to user names that start with 'be' (for example, betty_buyer)
user[%bu%]	Restricts operation to users with 'bu' in the name (for example, buyer and betty_buyer)

6. Click **Invoke**, located at the upper-right corner of the page, to proceed with the delete operation.
7. A confirmation message will display in the page. Click **Return** to return to the list of operations.

Deleting a Customization for a Specific Page for a Particular User

1. Navigate to the MDSAppRuntime management bean (MBean), as described in the section Using the System MBean Browser and the MDSAppRuntime MBean.
2. Delete metadata by selecting the deleteMetadata operation available from the Operations tab.
3. For excludeBaseDocs, select true. This is a Boolean value indicating whether to exclude base metadata documents from being deleted.
4. For docs, click the pencil icon. On the Edit Parameter popup, click **Add**, and in the Element box, enter a list of comma-separated, fully qualified document names or document name patterns.

   To delete all customizations for a user, enter "/**" (without the quotes).

This will recursively delete all customizations under "/" including any other customizations located in the folder(s) under it. Click **OK**.

5. For restrictCustTo, click the pencil icon. On the Edit Parameter popup, click **Add**, and in the Element box, enter the list of customization layer names. This can be used to restrict the operation to only customization documents that match the specified customization layers.

   Each customization layer name can contain, within a pair of brackets, optional customization layer values and value patterns separated by commas. Wildcards (%) may also be used for restricting the operation.

   For example:

Table 3–9 Customization Values

Value	Description
user[buyer]	Restricts operation to the user 'buyer'
user[buyer, betty_buyer]	Restricts operation to the users 'buyer' and 'betty_buyer'
user[bu%]	Restricts operation to user names that start with 'bu' (for example, buyer)
user[be%]	Restricts operation to user names that start with 'be' (for example, betty_buyer)
user[%bu%]	Restricts operation to users with 'bu' in the name (for example, buyer and betty_buyer)

6. Click **Invoke**, located at the upper-right corner of the page, to proceed with the delete operation.
7. A confirmation message will display in the page. Click **Return** to return to the list of operations.

Importing All Metadata Services Customizations

1. Navigate to the MDSAppRuntime management bean (MBean), as described in the section Using the System MBean Browser and the MDSAppRuntime MBean.
2. Import metadata by selecting the importMetadata operation available from the Operations tab.
3. For fromLocation, enter the path of the directory or archive from which the documents will be imported.

   Example: /tempDir/downloads/mdsExport/RetailAppsFrameworkTestMDS.zip
4. Click **Invoke**, located at the upper-right corner of the page, to proceed with the import operation.
5. A confirmation message will display in the page. Click **Return** to return to the list of operations.

Importing a Specific Page Customization for a User

1. Navigate to the MDSAppRuntime management bean (MBean), as described in the section Using the System MBean Browser and the MDSAppRuntime MBean.
2. Import metadata by selecting the importMetadata operation available from the Operations tab.

3. For fromLocation, enter the path of the directory or archive from which the documents will be imported.

   Example: /tempDir/downloads/mdsExport/RetailAppsFrameworkTestMDS.zip

4. For excludeBaseDocs, select true. A Boolean value indicating whether to exclude base metadata documents from being imported.

5. For docs, click the pencil icon. On the Edit Parameter popup, click **Add**, and in the Element box, enter a list of comma-separated, fully qualified document names or document name patterns.

   To import customizations for a specific page, simply enter the fully qualified base document name in the Element box.

   For example:
   /oracle/retail/apps/framework/uishell/skin/page/TestTablesAndTrees.jsff

   You can provide the path to multiple documents to be imported, by clicking the **Add**. When done, click **OK**

6. For restrictCustTo, click the pencil icon. On the Edit Parameter popup, click **Add**, and in the Element box, enter the list of customization layer names. This can be used to restrict the operation to only customization documents that match the specified customization layers.

   Each customization layer name can contain, within a pair of brackets, optional customization layer values and value patterns separated by commas. Wildcards (%) may also be used for restricting the operation.

   For example:

Table 3–10 Customization Values

Value	Description
user[buyer]	Restricts operation to the user 'buyer'
user[buyer, betty_buyer]	Restricts operation to the users 'buyer' and 'betty_buyer'
user[bu%]	Restricts operation to user names that start with 'bu' (for example, buyer)
user[be%]	Restricts operation to user names that start with 'be' (for example, betty_buyer)
user[%bu%]	Restricts operation to users with 'bu' in the name (for example, buyer and betty_buyer)

7. Click **Invoke**, located at the upper-right corner of the page, to proceed with the import operation.

8. A confirmation message will display in the page. Click **Return** to return to the list of operations.

Creating Metadata Labels

1. Navigate to the MDSAppRuntime management bean (MBean), as described in the section Using the System MBean Browser and the MDSAppRuntime MBean.

2. Create metadata label by selecting the createMetadataLabel operation available from the Operations tab.

3. For label, enter a valid name of the new metadata label to be created.

4. Click **Invoke**, located at the upper-right corner of the page, to proceed with the operation.
5. A confirmation message will display in the page. Click **Return** to return to the list of operations.

Promoting Metadata Labels

1. Navigate to the MDSAppRuntime management bean (MBean), as described in the section Using the System MBean Browser and the MDSAppRuntime MBean.
2. Create metadata label by selecting the promoteMetadataLabel operation available from the Operations tab.
3. For label, enter a valid name of the new metadata label to be promoted. Promoting metadata labels can be used to roll back to an earlier version of the document, as captured by the label.
4. Click **Invoke**, located at the upper-right corner of the page, to proceed with the operation.
5. A confirmation message will display in the page. Click **Return** to return to the list of operations.

Listing Metadata Labels

1. Navigate to the MDSAppRuntime management bean (MBean), as described in the section Using the System MBean Browser and the MDSAppRuntime MBean.
2. List all metadata labels by selecting the listMetadataLabels operation available from the Operations tab.
3. Click **Invoke**, located at the upper-right corner of the page, to proceed with the operation.
4. This will list all the available metadata labels previously created. You can use this, for example, to get the latest metadata label that you want to promote.
5. Click **Return** to return to the list of operations.

Deleting Metadata Labels

1. Navigate to the MDSAppRuntime management bean (MBean), as described in the section Using the System MBean Browser and the MDSAppRuntime MBean.
2. Delete metadata label by selecting the deleteMetadataLabel operation available from the Operations tab.
3. For label, enter a valid name of the metadata label to be deleted.
4. Click **Invoke**, located at the upper-right corner of the page, to proceed with the operation.
5. A confirmation message will display in the page. Click **Return** to return to the list of operations.

4

Security in Retail Applications

Retail Applications leverage ADF's security framework that is based on the Oracle Platform Security Services.

This section discusses the various assumptions around security for Retail Applications.

SSO Setup for Retail Fusion Platform Applications

Retail Fusion Platform provides the following applications as enterprise archive (EAR) files to Retail applications. By default, these applications are installing as part of Retail applications.

1. RetailAppsAdminConsole(RAAC)
2. RetailAppsMobileSecurity
    a. RetailAppsMobileBasicAuth
    b. RetailAppsMobileAccessService
3. RetailAppsRESTServices

In SSO environment, follow the SSO setup procedure for these applications similar to Retail applications.

Displaying External Application Contents in Non-SSO Environments

Retail Applications allow retailers to display content from external applications. These contents are typically business intelligence reports from a third party application that are configured to display within the Retail Application's dashboard.

Some of these contents might be secured requiring users to login before the contents can be accessed and displayed.

In non-SSO environments, when a user logs out of the Retail Application, they may not be logged out of any secured content they had configured access to. Therefore, it is highly recommended that retailers only configure access to external content in a SSO-enabled environments where the application logout manages the logout from any other secured content that was previously accessed.

5

Web Services in Retail Applications

Retailers can access backend functionality in Retail Applications by calling the applications' ReSTful Web Services.

Refer to **http://www.oracle.com/technetwork/articles/javase/index-137171.html** to learn more about ReST as an architectural style applied to building web services.

Common Characteristics of Retail Application ReSTful Web Services

This section describes the common characteristics.

Deployment

A Retail Application will package its ReST services as part of the application's Enterprise Archive (EAR) file. Specifically, those services are packaged as a Web Archive (WAR) within the EAR.

Installation of the ReST web services is therefore done by default.

Security

Services are secured using J2EE-based security model.

- Realm-based User Authentication. This verifies users through an underlying realm. The username and password is passed using Http Basic authentication.
- Role-based Authorization. This assigns users to roles, which in turn are granted or restricted access to resources/services. The authorization of ReSTful web services is static and cannot be reassigned to other rules post installation. The following role(s) is/are associated with ReSTful Web Services and should be added to the Enterprise LDAP:

  All enterprise roles defined above are mapped in web.xml and weblogic.xml of the ReST Service webapp.

The communication between the server and client is encrypted using one way SSL. In non-SSL environments the encoding defaults to BASE-64 so it is highly recommended that these ReST services are configured to be used in production environments secured with SSL connections.

Standard Request and Response Headers

Retail Application ReSTful web services have the following standard HTTP headers:

```
Accept: application/xml or application/JSON
Accept-Version: 16.0 (service version number)
```

```
Accept-Language: en-US,en;q=0.8
Accept-Versioning: False
```

> **Note:** Specify 'Accept-Versioning: False' in the Service Request to bypass 'Accept-Version' validation on services.

Depending on the type of the operation or HTTP method, the corresponding response header is updated in the Http response with the following codes:

- GET/READ : 200
- PUT/CREATE : 201 created
- POST/UPDATE : 204
- DELETE : 204

Standard Error Response

Example response payload in case of service error is depicted below:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<messagesRDOes>
 <messagesRDO>
   <message>REST Service Version Mismatch</message>
   <messageType>ERROR</messageType>
   <status>BAD_REQUEST</status>
 </messagesRDO>
</messagesRDOes>
```

- message: The error message - translated.
- messageType: Value of 'ERROR' is returned.
- status : For a bad request or error, the status is BAD_REQUEST.
- The http error code for an error response is 400.

List of ReSTful Web Services

This section covers the following:

Platform ReSTful Web Services

This section lists Web Services provided by the Retail Fusion Platform.

Notification ReST Services

Service endpoints have been enabled for the Notifications Framework. These endpoints are deployed separately and can be accessed by the following url:

http://server:port/RetailAppsReSTServices/services/private/Notifications/*

The endpoints exposed are detailed below.

Table 5–1 Notification ReST Services

Description	URL	HTTP Method
Creating Notifications	/Notifications/create	POST
Updating Notifications	/Notifications/update	PUT
Deleting Notifications	/Notifications/delete/{id}	DELETE
Fetch Notifications	/Notifications/fetch?appCode={appCode}	GET
Get number of unread Notifications	/Notifications/fetch/unreadCount?appCode={appCode}	GET
Search Notifications	/Notifications/search?appCode={appCode}	GET
Filter Notifications - Return list of Notifications	/Notifications/filter/list	POST
Filter Notifications - Return grouped list of Notifications	/Notifications/filter/group	POST
Filter Notifications - Return a summarized list of Notifications	/Notifications/filter/summarize	POST
Count Notifications matching the filter	/Notifications/filter/count	POST
Persist the Criteria	/Notifications/criteria	POST
Fetch the Criteria	/Notifications/fetch/criteria?appCode={appCode}	GET
Fetch Recipients	/Notifications/fetch/recipients/{id}	GET
Fetch Notification Context	/Notifications/fetch/context/{id}	GET
Fetch Notification Time Periods	/Notifications/fetch/timeperiods?appCode={appCode}	GET
Fetch Notification Hierarchy Levels	/Notifications/fetch/hierarchylevels?appCode={appCode}	GET
Fetch Notification Types	/Notifications/fetch/notificationtypes?appCode={appCode}	GET
Status update for multiple Notifications	/Notifications/update/multiple/status	PUT
Delete multiple Notifications	/Notifications/delete/multiple	POST

Access Control ReST Service

Table 5–2 Access Control ReST Service

Description	URL	HTTP Method
FetchFilteredRoles	/AccessRoles	POST

Favorites ReST Services

Table 5–3 Favorites ReST Services

Description	URL	HTTP Method
Fetch all Favorites	/Favorites/fetch	GET
Add a Favorite	/Favorites/add	POST
Delete a Favorite	/Favorites/delete	DELETE
Update a Favorite	/Favorites/update	POST
Replace a Favorite	/Favorites/replace	POST

REIM ReSTful Web Services

Several services used by the mobile application are exposed via RESTful web services.

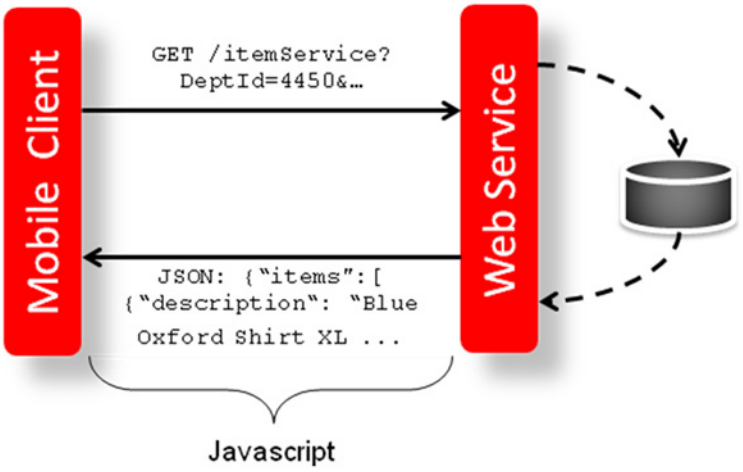


Table 5–4 RESTful Web Services

Service Operation	Type	URL
Dashboard	Get	/dashboard
Supplier List	Get	/suppliers?employeeIds=[employeeId Array]&specificSearchValue=[]&numberOfInvoices=[]&supplierIds=[supplierId Array]&sortOrder=[]&screenSort=[]&pageSize=[]&pageNumber=[]
Supplier	Get	/supplier?supplierId=[]
Invoice List	Get	/invoices?employeeIds=[employeeId Array]&invoiceAmountHigh=[]&cashDiscount=[]&screenSort=[]&pageSize=[]&costDiscrepancy=[]&unitDiscrepancy=[]&specificSearchValue=[]&supplierIds=[supplierId Array]&pageNumber=[]&daysOut=[]&invoiceAmountLow=[]
Employee List	Get	/employees?pageNumber=[]&pageSize=[]&screenSort=[]&sortOrder=[]

Table 5–4 (Cont.) RESTful Web Services

Service Operation	Type	URL
Employee	Get	/employee?employeeId=[]
Supplier Criteria	Get	/supplierCriteria?pageNumber=[]&pageSize=[]&specificSearchValue=[]
Employee Criteria	Get	/employeeCriteria?pageNumber=[]&pageSize=[]&specificSearchValue=[]

6

In-Context Launching Task Flows in Retail Applications

Retail Applications can expose select task flows retailers can directly launch into. This feature is referred to as in-context launching in a Retail Application.

Retailers can launch these task flows directly through specific URLs.

Retail Applications will provide information about the various task flows retailers can in-context launch into including the URLs and the required parameters.

Retailers can use these URLs in other web pages as links. For example: the URL to a task flow that invokes the Summary Match Flow in a Retail Application can be added as a link to dashboard report on a BI server.

When the user clicks on a URL to a task flow, the Retail Application will open in a new browser window or tab depending on the specified target of the URL. The requested task flow will be shown as a UI tab within the Retail Application.

Limitations of In-Context Launch via URLs

- The In-Context launch feature will not detect if there is an already opened window or tab for the Retail Application. So when a user clicks on a link to a Retail Application's task flow, say, a report on a BI server, a new browser window or tab will be opened even though the user already has an existing browser window or tab opened for that same Retail Application.
- If a BI dashboard is added on the Retail Application, and if that dashboard has a report that contains links to in-context launchable task flows in the same Retail Application, a new browser window or tab will still be opened.

List of In-Context Launch Task Flows

Summary Match Flow

In-context launch from the Invoices Due Soon report located on the user dashboard. When this screen is launched from the Invoices Due Soon panel on the dashboard, the system fetches all the invoices and receipts associated with the default match key from the supplier and for the given invoice.

http://<host>:<port>/ReimViewController/faces/Home?navModelItemId=SummaryMatchService&supplier=130001&invoice=INVC 51319 02

Table 6–1 Summary Match Flow

Parameter ID	Required?	Description	Sample URL
Supplier	Yes	Supplier/Vendor Id	http://<host>:<port>/ReimView Controller/faces/Home?navMod elItemId=SummaryMatchService &supplier=130001&invoice=INVC 51319 02
Invoice	Yes	Invoice/Document Id	http://<host>:<port>/ReimView Controller/faces/Home?navMod elItemId=SummaryMatchService &supplier=130001&invoice=INVC 51319 02

Discrepancy List Flow

In-context launch from the Invoices Due Soon report located on the user dashboard. When this screen is launched from the Invoices Due Soon panel on the dashboard, the system fetches all the discrepant invoices for the given supplier and invoice

http://<host>:<port>/ReimViewController/faces/Home?navModelItemId=Discrepa ncyListService&supplier=2300&invoice=am_6901_1

Table 6–2 Discrepancy List Flow

Parameter ID	Required?	Description	Sample URL
Supplier	Yes	Supplier/Vendor Id	http://<host>:<port>/ReimView Controller/faces/Home?navMod elItemId=DiscrepancyListService &supplier=2300&invoice=am_ 6901_1
Invoice	Yes	Invoice/Document Id	http://<host>:<port>/ReimView Controller/faces/Home?navMod elItemId=DiscrepancyListService &supplier=2300&invoice=am_ 6901_1

7

Customizing Retail Applications

This chapter discusses supported steps for customizing Retail applications.

Prerequisite Concepts

Retailers must understand the following concepts before performing any of the supported customization scenarios discussed in this chapter.

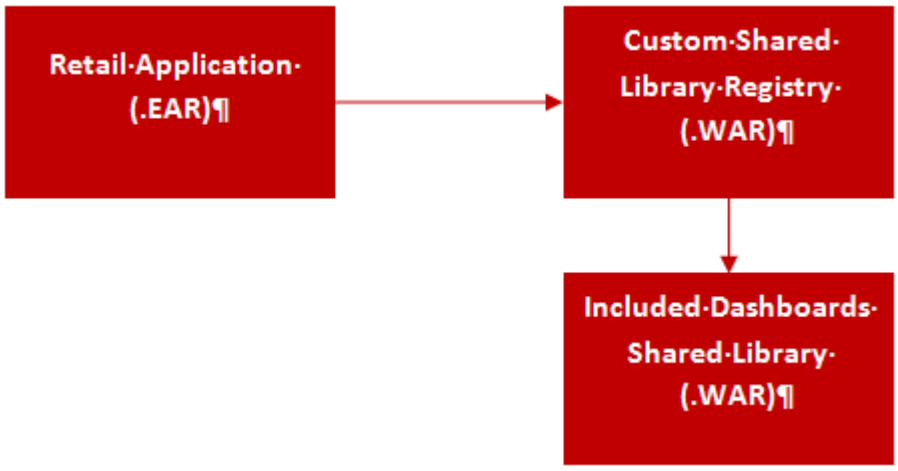
- Understanding the Deployment of Retail Applications
- Understanding the Retail Application User Interface

Understanding the Deployment of Retail Applications

In order to customize the Retail Application, retailers must understand the deployment of the Retail Application.

A Retail Application can install one or more application artifacts into a WebLogic server instance. The diagram below shows a typical installation of a Retail Application:

Figure 7–1 Retail Application Typical Installation



The Retail Application is deployed as a standalone application and deployed using an EAR file.

Two shared libraries are typically installed along with the application EAR:

- The Included Dashboards Shared Library
- The Custom Shared Library Registry

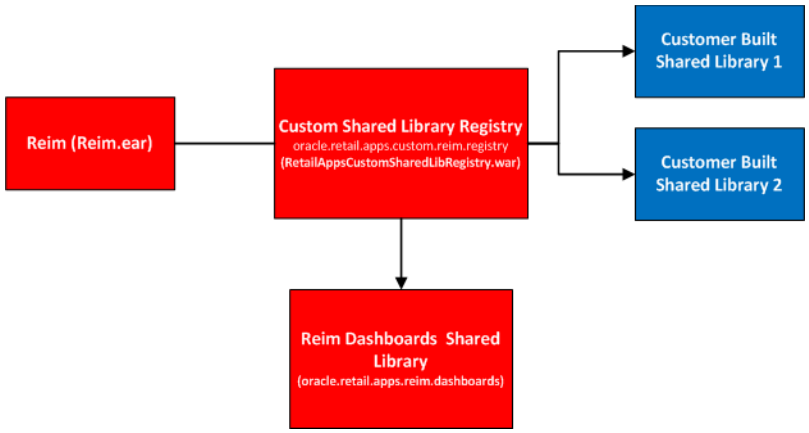
The Included Dashboards Shared Library contains the Retail Application-built dashboards. These dashboards can be customized. Further discussion on Retail Application Included Dashboards can be found in the section, Understanding Dashboards in Retail Applications.

A Retail Application that allows for customization will have as part of its installation an intermediary shared library that will serve as a registry to reference the actual retailer-built shared libraries. This is called the Custom Shared Library Registry.

When retailers need to add their own content into the application, metadata to register their content into the application's UI including the binaries for the content itself (e.g. task flows and pages) are expected to be packaged into a Web Archive (WAR) file and deployed as a shared library in the same managed server as the application itself.

Then, the names of these shared libraries have to be referenced in the Custom Shared Library Registry.

Figure 7–2 Custom Shared Library Registry

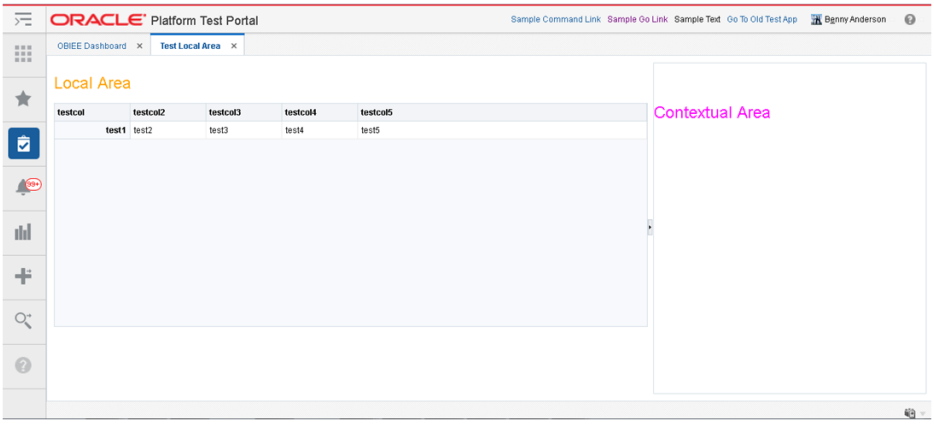


Understanding the Retail Application User Interface

Retail Applications allow retailers to add and display custom content such as new workflows and dashboards. These custom contents can be accessed from the Retail Application's user interface (UI).

The Retail Application user interface organizes the contents into visual containers that fulfill common layout and navigational requirements in a structured, consistent manner.

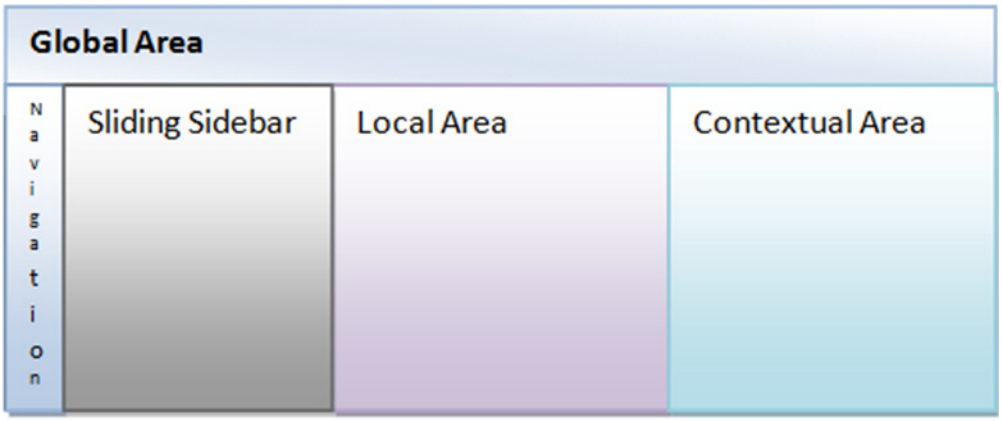
Figure 7–3 Retail Application User Interface



The high-level containers or areas in the UI are highlighted below:

- Global Area
- Navigation Pane and Sliding Sidebar Menu
- Local Area
- Contextual Area

Figure 7–4 User Interface Containers



The Local Area is the main content area of the application. Workflows that allow users to get tasks done in the application are presented in this area. The workflows are launched into this area as a result of actions the user take in the navigation pane, sliding sidebar panel and global area.

The Navigation Pane organizes the different actions a user can take on the application as iconic menu items. The options or actions available to the users for each menu item is presented in the Sliding Sidebar panel next to the Navigation Pane. There are five (5) standard Navigation Pane menus that are available to users:

- Application Navigator/Switcher – This menu allows users to switch between Retail Applications.
- Favorites – This menu allows users to access their own favorite list of actions or tasks within the application.

- Tasks – This menu presents a hierarchy of tasks or actions the user can access in the application.
- Reports – This menu presents a hierarchy of links to the application reports and dashboards.
- Notification – This menu allows users to access unread and past notifications around business processes that are happening in the application.

The Global Area of the application contains branding information on the left hand side and application wide menu options on the right hand side. Application wide menu options include access to application preferences, login and logout, application help and application information.

The Contextual Area is a collapsible area on the right of the page, which provides space to present information that can assist users in completing their tasks. The Contextual Area is presented per Local Area tab. Each task flow is presented in the local area can have a contextual area.

Supported Customization Scenarios

When retailers want to add new content such as new pages, business components, or even Java code into a Retail Application, retailers will need to create a custom shared library, deploy that into the same managed server as the Retail Application, and register that library into the Retail Application.

Adding a Custom Shared Library

As discussed in the section, Understanding the Deployment of Retail Applications, a Retail Application that allows for customization will have as part of its installation an intermediary shared library that will serve as a registry to reference the actual retailer-built shared libraries. This is called the Custom Shared Library Registry.

This section contains steps retailers can follow to create a custom shared library that will contain new custom made content, reference that shared library from the registry, and deploy it to the server.

Downloading JDeveloper

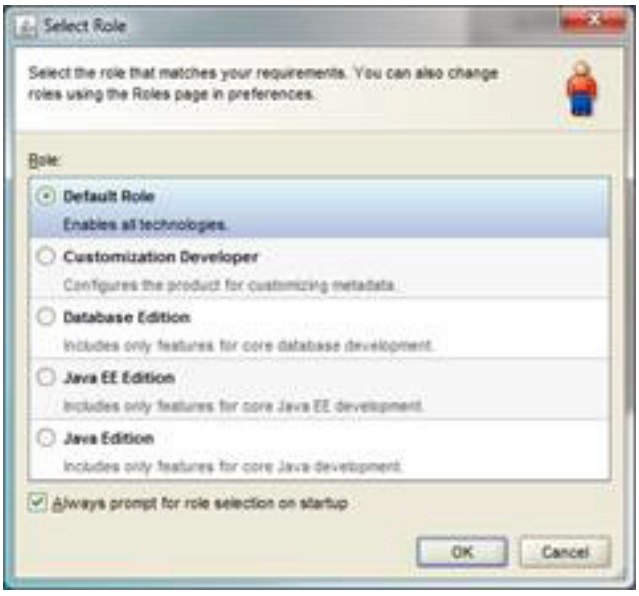
To create the custom shared library, it is recommended that retailers download and install JDeveloper version 12.2.1 by following the link below:

http://www.oracle.com/technetwork/developer-tools/jdev/downloads/index.html

Creating the Custom Shared Library Workspace through JDeveloper

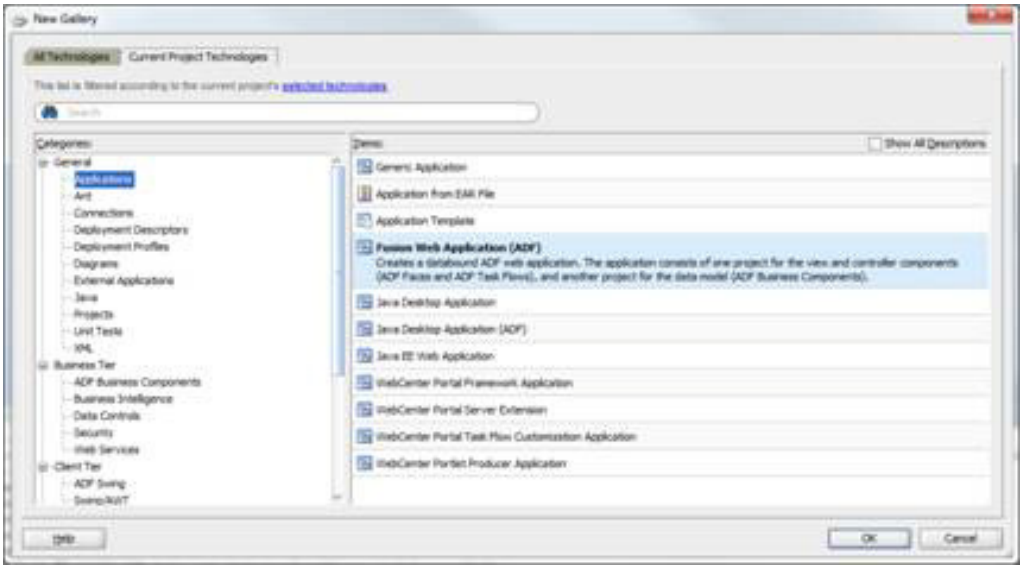
This section describes the steps to create and configure the Custom Shared Library Workspace in JDeveloper, which will house the code for any custom content the retailer wants to add into the Retail Application. Through JDeveloper, a shared library web application archive file (.WAR) can be generated which can be deployed in the same managed server as the Retail Application.

1. Open JDeveloper and choose **Developer Role** when prompted.

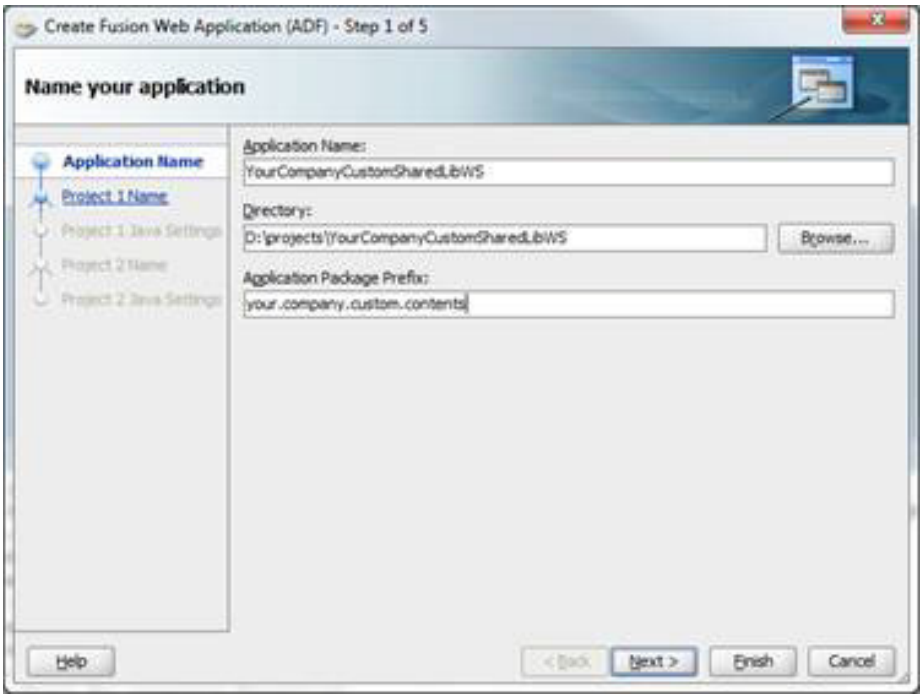


2. Create a new Fusion Web Application JDeveloper workspace.

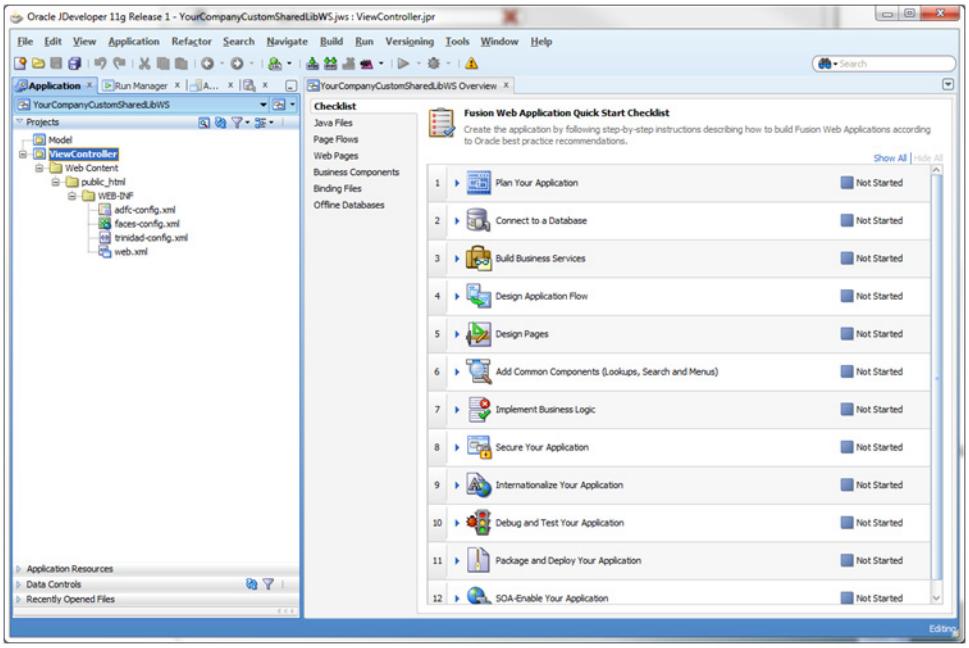
   a. Go to **File > New** to invoke the New Gallery dialog. Choose **Fusion Web Application (ADF)** as the type of application to create and click **OK**.



   b. Provide a meaningful application name, a directory path to the workspace, and an application (Java) package prefix. Click **Finish**.

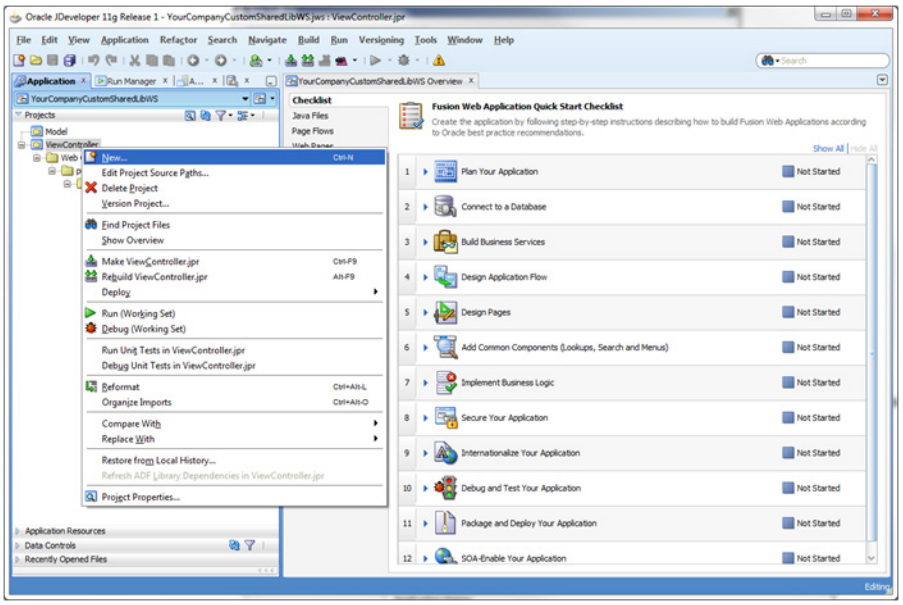


c. JDeveloper will generate a new workspace with two projects: Model and View-Controller.

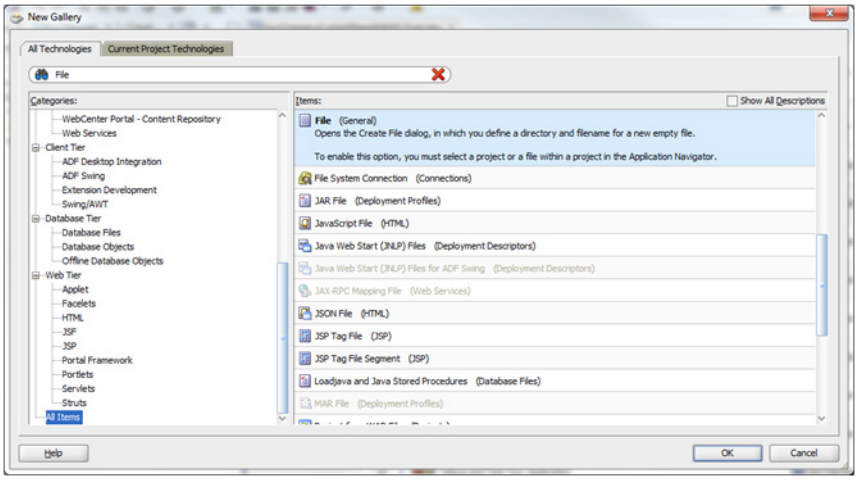


3. Add a Manifest file containing the name of the shared library.

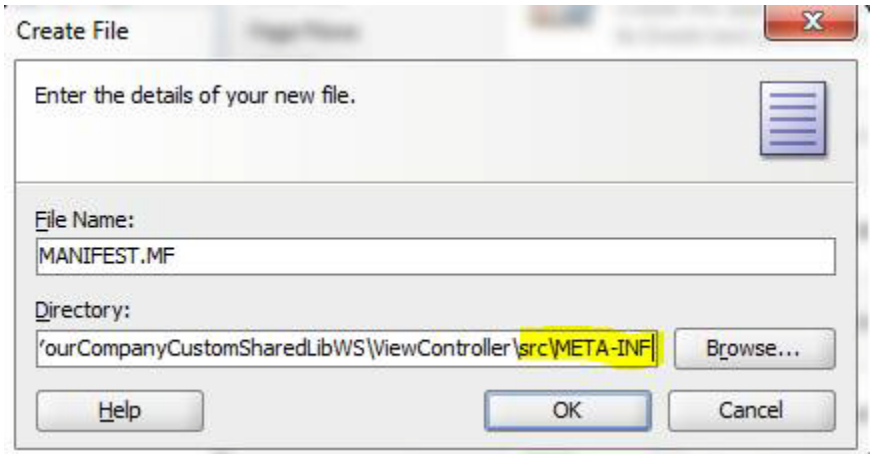
a. Right-click on the **View-Controller** project and choose **New** from the Context menu.



b. The New Gallery dialog appears. Choose the option, **File (General)**, in the All Technologies section of the dialog. Click **OK**.



c. The Create File dialog opens. For the **File Name**, specify MANIFEST.MF. For the **Directory**, the new file **must be added under** the src/META-INF **sub-directory** under the View-Controller project's directory.



d. Edit the new MANIFEST.MF file and add the following entries:

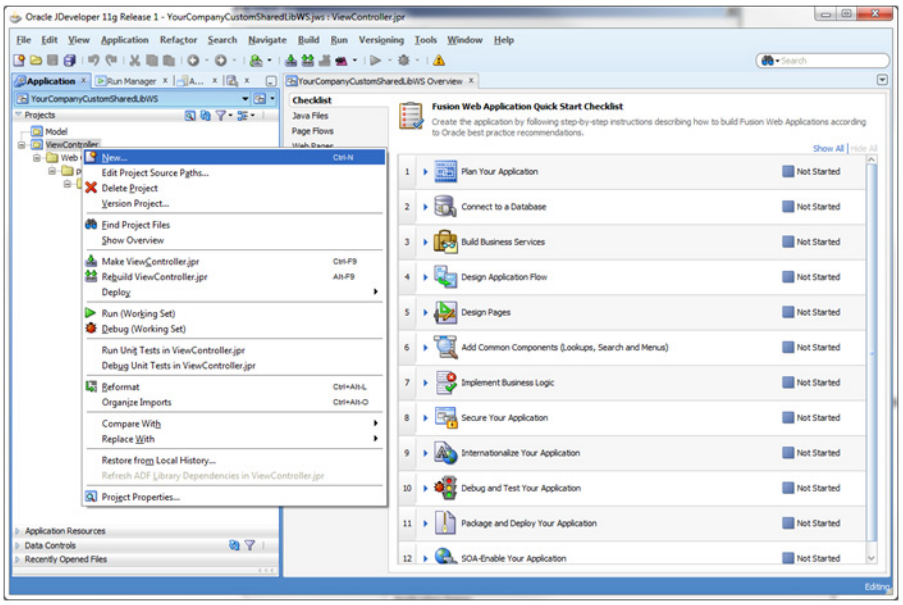
```
Manifest-Version: 1.0
Implementation-Vendor: companyName
Implementation-Title: Custom Shared Library for companyName
Implementation-Version: 1.0
Extension-Name: companyname.custom.shared.lib
Specification-Version: 1.0
Created-By: companyName
```

Modify the contents such that meaningful and unique values are used for *Implementation-Vendor*, *Implementation-Title*, *Extension-Name*, and *Created-By*. Example:

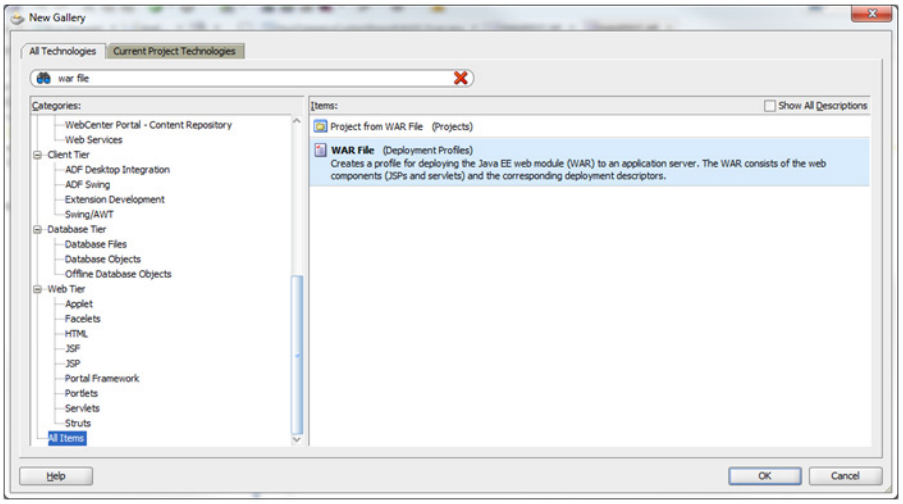
```
Manifest-Version: 1.0
Implementation-Vendor: Acme Retail
Implementation-Title: Custom Shared Library for Acme Retail
Implementation-Version: 1.0
Extension-Name: acmeretail.custom.shared.lib.procurement
Specification-Version: 1.0
Created-By: AcmeRetail
```

4. Create a deployment profile for the shared library.

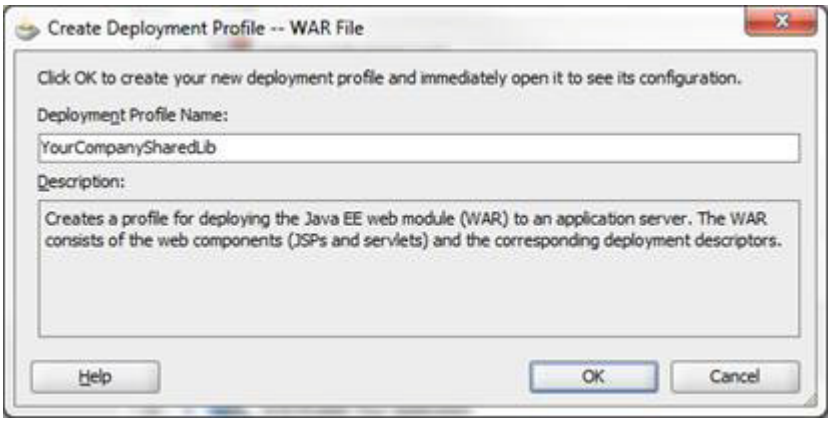
   a. Right-click on the **View-Controller** project and choose **New** from the Context menu.



b. The New Gallery dialog appears. Choose the option, **WARFile (Deployment Profiles)**, in the All Technologies section of the dialog. Click **OK**.



c. Provide a unique and meaningful name for the Deployment Profile and click **OK**.

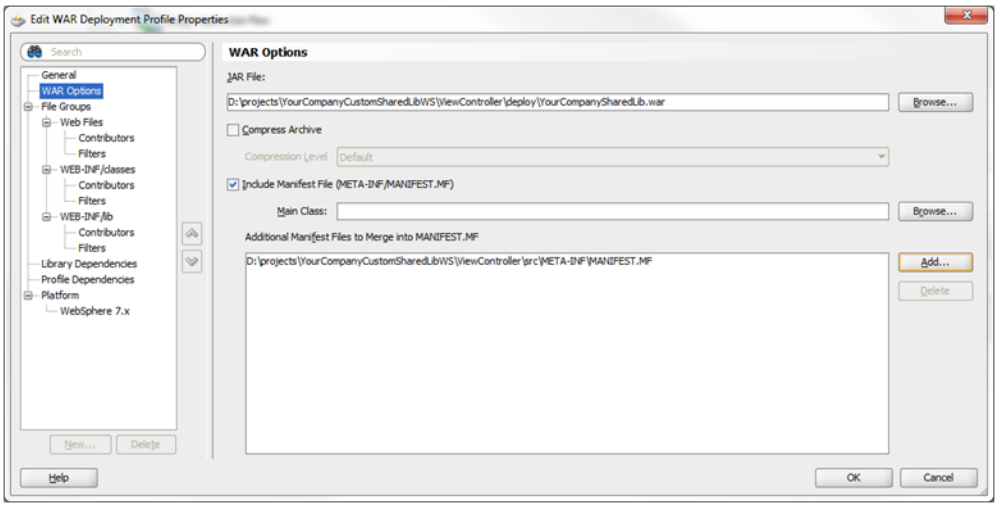


d. The Edit WAR Deployment Profile Properties dialog is shown.

e. Under the General section, make sure that the **Specify Java EE Web Context Root** is selected without any value.

When you navigate away from this section, you will be prompted to confirm that you really want a blank context root. Click **Yes** to the confirmation dialog.

f. Under the WAR Options section, **enable** the **Include Manifest File** option and **Add** the MANIFEST.MF file you created under …/View-Controller/src/META-INF.

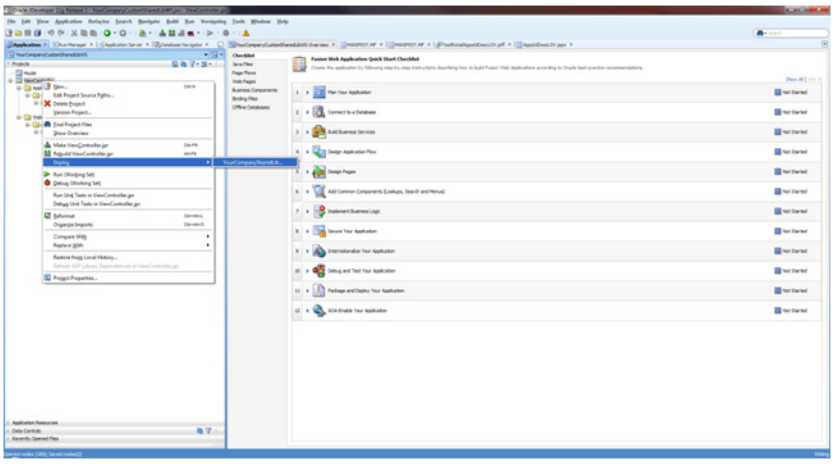


g. Click **OK**.

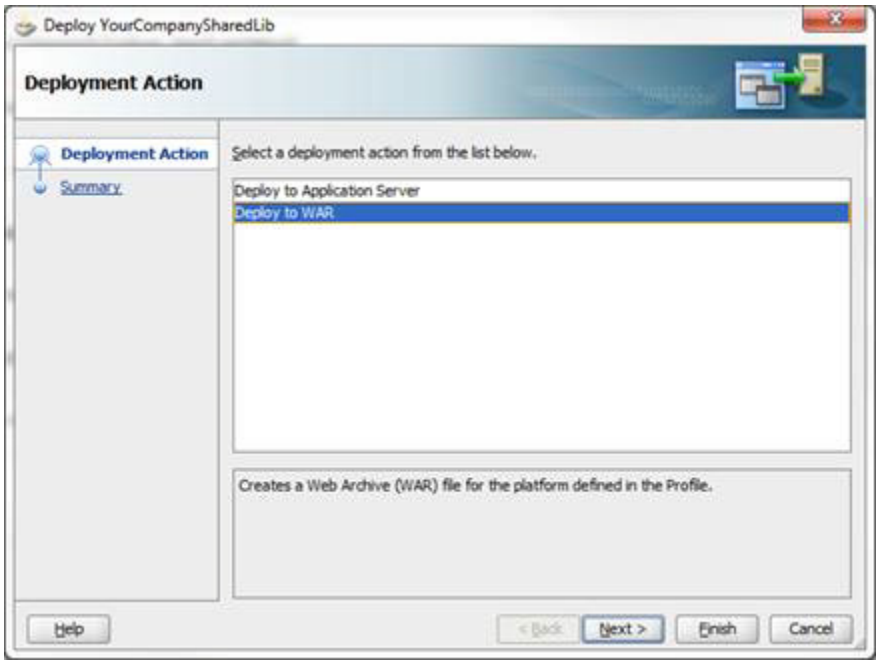
Generating and Deploying the Custom Shared Library WAR

This section describes the steps on how to generate and deploy the Custom Shared Library WAR file:

1. Generate the share library WAR file through JDeveloper.
    a. Open the Custom Shared Library workspace.
    b. Right-click on the **View-Controller** project and choose the shared library WAR deployment profile created previously under the **Deploy** option.



    c. The Deploy dialog opens. Select **Deploy to WAR** and click **Finish**.

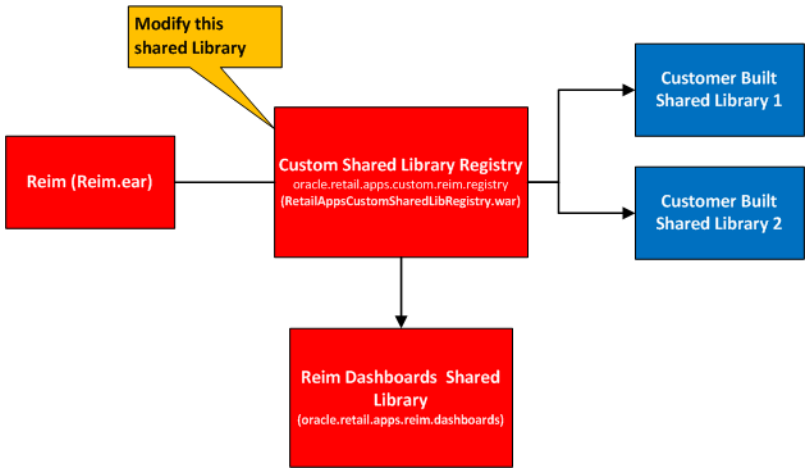


    d. JDeveloper generates the WAR file into the View-Controller project folder sub-directory, deploy.

2. Deploy the generated WAR file to the same managed server as the Retail Application as **a shared library**. Refer to section 6 of the Deploying Applications to Oracle WebLogic Server documentation (**http://docs.oracle.com/cd/E23943_01/web.1111/e13702/deploy.htm**). This task has to be done using the WebLogic Administration Console or Enterprise Manager Fusion Middleware Control with a user having WebLogic administrator permissions.

Referencing the Custom Shared Library from the Retail Application

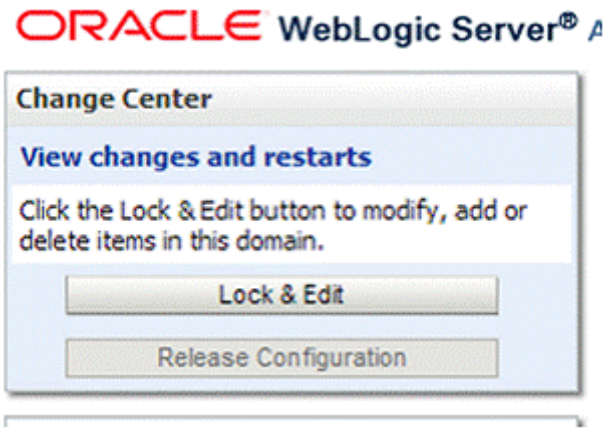
Once the retailer has created and deployed its own custom shared library, the retailer needs to modify the configuration of the Custom Shared Library Registry to add references to the retailer's shared library.



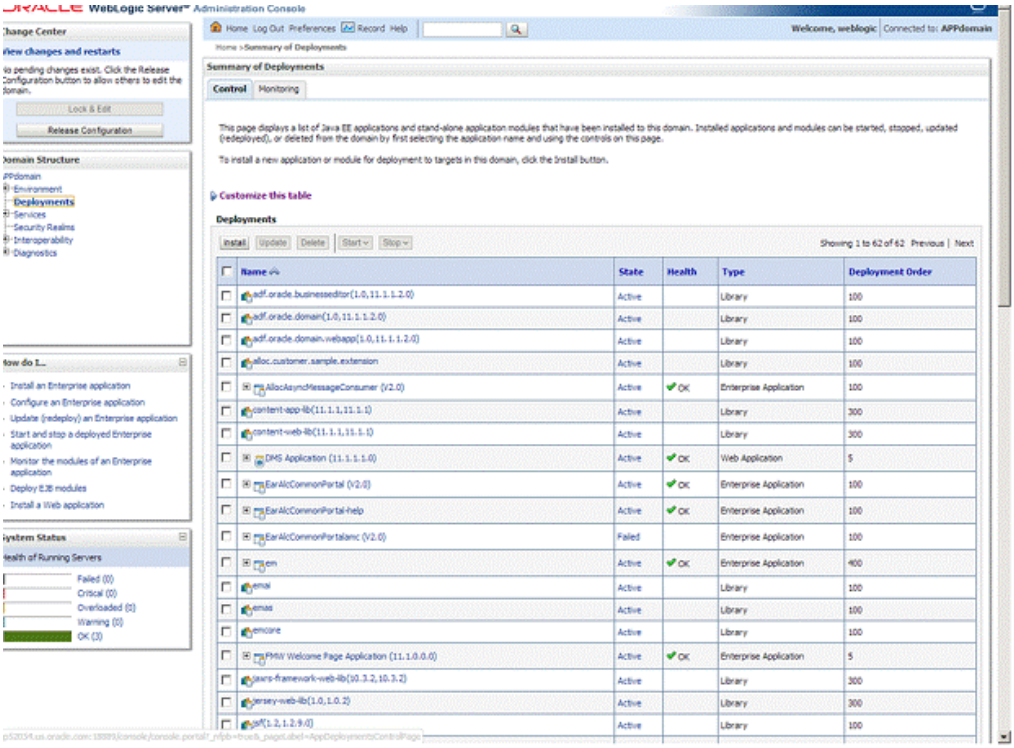
oracle.retail.apps.custom.reim.registry

To do this:

1. Log into the WebLogic Administration Console as a user with administrative permissions.
2. If the Administration Console was configured with domain configuration locking, go ahead and click **Lock & Edit** to ensure that other administrators can be prevented from making changes during your edit session.

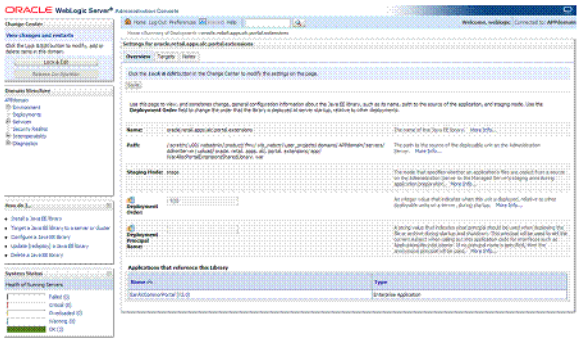


3. Navigate to the Deployments section.



4. Look for the Retail Application deployment and shut it down. Choose **Force Stop Now** when appropriate. Wait for the shutdown process to complete.

5. Get the deployment location of the Retail Application's custom shared library registry. Under the deployments list, click on the link for the library named **`oracle.retail.apps.alc.portal.extensions`**. The Settings page for the library appears.



The Settings page will show the file location of the registry's WAR file under the Path entry.

Make a note of this file location.

6. Using the operating system's file manager application, go to the location of the WAR file. You need read and write permissions to the file system where the WAR file is located.
7. Make a copy of the WAR file as back-up.
8. Open the original WAR file using an archive file manager and update the /WEB-INF/**weblogic.xml** by adding a new <library-ref> entry pointing to your custom shared library.

```
<library-ref>
<library-name>companyname.custom.shared.lib</library-name>
</library-ref>
```

> **Note:** The library-name has to match the Extension-Name you provided in your custom shared library's MANIFEST.MF file.

Once this change is done, you have now linked your custom shared library to the Retail application.

9. Return to the WebLogic Administration Console. Navigate to **Environments' Servers** section. Under the Control tab, select the managed server where the Retail Application is deployed to. Shut it down and start it back up again.

Creating New ADF Contents

Custom ADF and Java based components can be coded into the Custom Shared Library workspace. Typically, retailers can include task flows that add new UI workflows into the Retail Application UI. These task flows may implement:

- A new workflow to maintain the Retail Application's data.
- A dashboard page to highlight key performance indicators and important metrics based on the Retail Application's data.

This section contains important points that developers must consider when building new custom ADF-based contents for Retail Applications.

Custom Shared Library Must Be Regenerated

Custom ADF content must be implemented within the custom shared library workspace. Retailers must create a new custom shared library workspace if none exists.

The WAR artifact for the custom shared library must be regenerated and redeployed. See Adding a Custom Shared Library for additional information.

New Components Should Have Security Grants

Since Retail Applications are secured web applications, the addition of new ADF UI pages and task flows will need to be secured using the application roles and policies that are recognized by the Retail Application. The provisioning process is done using the Oracle Enterprise Manager Fusion Middleware Control console.

Applying ADF Best Practices

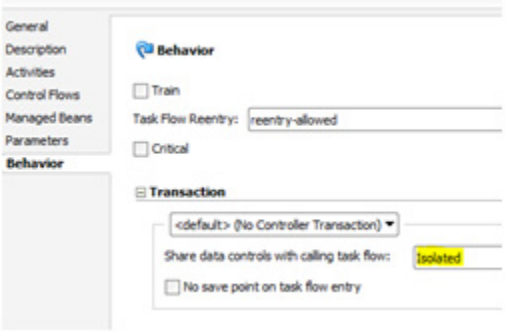
Developers will be building components in both Model and View-Controller layers using the tools and standards available for Oracle ADF. New ADF screens will have their own business components, task flows, pages, and bindings. These components must comply to published best practices and standards for that technology.

Refer to the Oracle JDeveloper and Oracle ADF Technical Resources web site (**http://www.oracle.com/technetwork/developer-tools/jdev/learnmore/index.html**) . Of particular interests are coding guidelines published under the ADF Architecture Square section of the site (**http://www.oracle.com/technetwork/developer-tools/adf/learnmore/adfarchitect-1639592.html**).

Task Flow and Page Configuration Must Be Supported

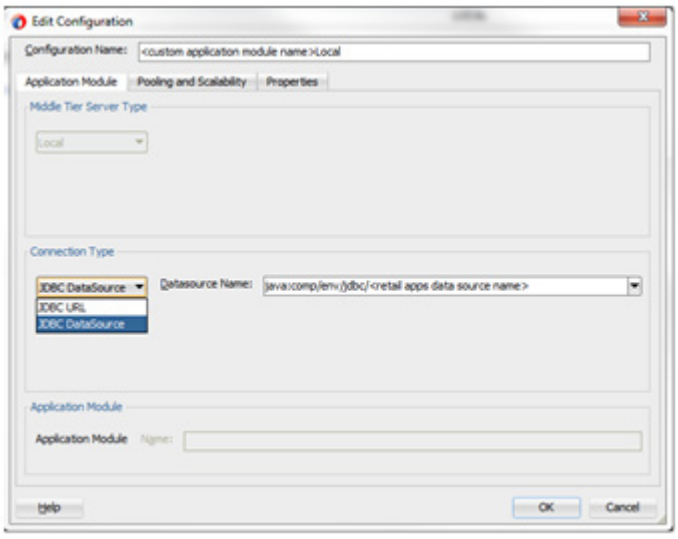
Task flows must be built as bounded task flows using JSF page fragments.

The task flow must have an isolated data control scope. This is configured in JDeveloper's task flow designer by going to the Behavior tab and setting data control share option as "Isolated".



The Same Data Source Must Be Used

New application modules must be configured to use Data Sources and not JDBC URLs when accessing the database layer. This can be configured in JDeveloper's application module designer.



The data source name must be the same as the Retail Application's primary data source name.

jdbc/AllocationApplicationDBDS

Adding or Modifying an Item in the Reports Menu

To add new contents in the Retail Application's Reports Menu, the retailer must override the reports menu model.

All items seen by the user in the Reports Menu are registered in an XML file called the Reports Menu Model.

This file is kept in the application's EAR file but can be extracted, copied and modified so retailers can add references to their own dashboards or reports.

To add or modify an item in the Reports Menu, follow the steps below:

1. Add a custom shared library. See Adding a Custom Shared Library for additional information.
2. Using JDeveloper, open the Custom Shared Library workspace in Developer Role.
3. Add new custom reports as required.
4. Regenerate the shared library WAR file from the workspace and redeploy the shared library. Shutdown and restart of the Retail Application and its shared library registry is required.
5. Changes Required in Shared Library Registry for Custom Reports Menu Model:
    a. Obtain the application's Reports menu model XML file.

       Reports Menu Model XML – Reim.ear\ReimViewController.war\WEB-INF\classes\oracle\retail\apps\framework\uishell\config\custom\HomeReportsMenuModel.xml

    b. Please refer the Section Reference the Custom Shared Library from the Retail Application., to register custom reports menu model shared library within the Retail Application.

    c. Open the original WAR file using an archive file manager and copy the above Reports menu model xml file in View-Controller project src directory, preferably under a subdirectory called custom.

d. Add a new or modify existing file called PageTemplateOverrideModel.properties under the View-Controller/src directory.

Modify this file and add the following entry:

```
Home.reportsMenuModelXml=<path to sidebar model xml within
view-controller/src>
```

Example:

```
Home.reportsMenuModelXml=/custom/HomeReportsMenuModel.xml
```

e. Modify the copy of the model. Add new items , which are created in the shared library project. Refer to the section, Reports Menu Model XML Items.

6. Test the Retail Application.

Reports Menu Model XML Items

The Reports menu model is composed of Item elements. An item renders as either a launchable link or a submenu in the Reports menu of the Retail Application. Below is an example of what a Reports Menu Model XML might look like:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<NavigationDefinition … >
    <Items>
       <Item id="myFolder" title="My Folder"  type="folder">
           <Items>

                <Item id="myContent" type="taskflow"
                      title="My Content">
                    <url>

/WEB-INF/mycompany/CustomDashboardTaskFlow.xml#CustomDashboardTaskFlow
                    </url>
                    <Parameters>
                        <Parameter id="productName">"Acme"</Parameter>
                    </Parameters>
                </Item>
           </Items>
       </Item>
    </Items>
</NavigationDefinition>
```

Item Attributes

Attribute	Description
visible	Indicates if the item should be rendered (visible) or not. It can be an EL Expression that evaluates to true or false. Or it can be a plain string value equal to "true" or "false". Defaults to "true".
type	Indicates the type of the item. The main values are "folder","taskflow","link". The type "folder" indicates that the item contains additional sub-navigation items. The type "taskflow" indicates it is a task flow, and the type "link" is used for URLs.
title	The attribute title is used to provide the title of the sidebar item. This attribute is a required attribute.

Attribute	Description
targetTitle	The attribute targetTitle is used to provide the title of the tab in the content area. This attribute is an optional attribute. If not provided, the value of the title attribute will be used as the title of the tab in the content area.
accessKey	The attribute accessKey is used to specify the keyboard keystroke or a group of keystrokes that are used to access the navigation item using the keyboard.
shortDesc	The attribute shortDesc is used to provide the description of the navigation item which will be displayed when the user hovers over a menu item.
showCloseIcon	Mainly used when the content is being rendered in a popup, and indicates if the 'x' icon should show or not on the top right corner of the popup to facilitate manual closing of the popup. Takes the values "true" or "false". Defaults to "false".
defaultContent	Indicates whether the corresponding items from the model show up by default in tabs in the content area when the page template loads. Takes values "true" or "false", or may even take an EL Expression evaluating to "true" or "false".
reuseInstance	The attribute reuseInstance is used to specify whether the navigation items with the same title and url will use the same tab or not. When this attribute is set to false, the request to open the content for that navigation item will always use a new tab. When the attribute is true, the navigation items with the same title and url will share the same tab in the content area. It defaults to true.
keysList	Takes name value pairs separated by a semicolon. The attribute keysList provides a way to differentiate two navigation items with the same title and url. If keysList is not provided, then the two navigation items with the same title and url will always use the same tab. When the keysList is provided, it provides uniqueness to the navigation items with the same title and url enabling them to use different tabs. Example keysList="key1=value1;key2=value2"
urlRendererHeight,	Example values are shown below. These used in the case of a popup and indicate the height and width of the popup.
urlRendererHeight, urlRendererWidth	Example values are shown below. These used in the case of a popup and indicate the height and width of the popup. urlRendererHeight="200px" urlRendererWidth="200px"
reloadTab	When this attribute is set to true, an already opened tab will be reloaded with the new input parameters for the taskflow. When it is set to false, a previously opened tab will only be re-focused, but not reloaded with new input parameters for the taskflow. The reload tab functionality has some limitations. Please see section 'Error! Reference source not found.

Attribute	Description
refreshOnDisclosure	When the navigation item has refreshOnDisclosure attribute set to true then the tab displaying that item in the Content Area will be refreshed every time it's disclosed. The attribute can take either of the two values true or false. Default is false. The attribute is useful in the scenarios where we want to display to the user the latest information from the database every time he/she comes back to the tab. The attribute should be used with caution because if the data in that tab is not committed before leaving the tab then the uncommitted data will be lost upon coming back to the tab.
dynamicTabFocus	When a navigation item is invoked, the tab displaying that item will have its text focused. To override this behavior, set dynamicTabFocus to "false". This attribute defaults to "true".
popupId	Applicable only when target="_popup". Must be a number between 1 and 15. This attribute allows consuming applications to target a specific popup within the UI Shell. The framework provides 15 popups that consuming applications can take advantage of. In case this attribute has not been specified, a default popup will be used by the framework. This default popup will not store its dimensions in MDS.
popupContentHeight	Applicable only when target="_popup". This attribute is used to provide the height in pixels of the resulting popup dialog window.
popupContentWidth	Applicable only when target="_popup". This attribute is used to provide the width in pixels for the resulting popup dialog window.
popupStretchChildren	Applicable only when target="_popup". This attribute is used to indicate the stretching behavior for the contents of the resulting popup window. The contents are referred to as child components. Valid values are: ■ none - does not attempt to stretch any children (the default value and value you need to use if you have more than a single child; also the value you need to use if the child does not support being stretched). ■ first - stretches the first child (not to be used if you have multiple children as such usage will produce unreliable results; also not to be used if the child does not support being stretched).
popupResize	Applicable only when target="_popup". This attribute is used to indicate the resulting popup window's resizing behavior. Valid values are: ■ on - user can resize the dialog with their mouse by dragging any of the dialog edges. ■ off - the dialog automatically sizes its contents if popupStretchChildren is set to "none"

Attribute	Description
popupHelpTopicId	Applicable only when target="_popup". This attribute is used to look up a topic in a helpProvider for the resulting popup window.
popupShortDesc	Applicable only when target="_popup". This attribute is used to provide short description of the resulting popup window.
contentListener	Specifies the fully qualified name of the class implementing the ContentListener interface. This allows applications to have the ability to inject any session or request values before opening tabs.
tabShortDesc	Specifies the text to be shown when the user hovers over the application tab. Using this attribute application team can keep the title short and the tabShortDesc as fully qualified tab name which can be shown as the tooltip of the tab. This attribute will be displayed as tab title in screen reader mode.

Item Sub-elements

Sub-element	Description
url	The location of the item being launched. If the type is "taskflow" - then the URL element must contain the path to the task flow XML. If the type is "link" - then the URL of the external system must be indicated in this subelement. The entire URL must be marked as character data (e.g. enclosed in CDATA)
Parameter	The <Parameters> sub-element within <Item> should list all the parameters to the dashboard page if there are any. Each parameter is represented as a <Parameter> element inside <Parameters>. The <Parameter> id should be the actual parameter reference name recognized by the dashboard URL. The value of each <Parameter> is a string value. This is the only supported data type.

Securing Access to Items

To restrict access to report items to specific security roles, set the visible property on the <Item> element for the dashboard URL to an Expression Language (EL) expression that calls ADF's securityContext API's isUserInRole method. Example:

```
<Item id="myDashboard1"
      type="link"
      title="Profitability Dashboard"
      visible="#{securityContext.isUserInRole['BUYER_JOB']}" >
```

The parameter to the securiytContext.isUserInRole method is a logical security role that is configured for the Retail Application. The API returns true if the user is included in the specified security role. If the user is not authenticated or is not found in the role, the API returns false.

Adding or Modifying an Item in the Tasks Menu

To add new contents in the Retail Application's Tasks Menu, the retailer must override the tasks menu model.

All items seen by the user in the Tasks Menu are registered in an XML file called the Tasks Menu Model.

This file is kept in the application's EAR file but can be extracted, copied and modified so retailers can add references to their own workflows.

To add or modify an item in the Tasks Menu, follow the steps below:

1. Add a custom shared library. Refer to the section, Adding a Custom Shared Library for details.
2. Using JDeveloper, open the Custom Shared Library workspace in Developer Role.
3. Add all the custom task menu items as required.
4. Regenerate the shared library registry WAR file from the workspace and redeploy the shared library. Shutdown and restart of the Retail Application and its shared library registry is required.
5. Changes Required in Shared Library Registry for Custom Task Menu Model
   a. Obtain the application's Tasks menu model XML file.

      Tasks Menu Model XML – /oracle/retail/apps/framework/uishell/config/custom/HomeTaskMenuModel.xml

   b. Please refer the section Reference the Custom Shared Library from the Retail Application, to register custom task menu model shared library with in the Retail Applications.

   c. Open the original WAR File using an archive file manager and copy the above TaskMenuModel xml file in the View-Controller project src directory, preferably under a subdirectory called custom.

   d. Add a new file called PageTemplateOverrideModel.properties under the View-Controller/src directory. Modify this file and add the following entry:

      ```
      Home.taskMenuModelXml=<path to sidebar model xml within
      view-controller/src>
      ```

      Example:

      ```
      Home.taskMenuModelXml=/custom/HomeTaskMenuModel.xml
      ```

   e. Modify the copy of the model. Add new itemsdefined in the custom shared library above. The format of the tasks menu model is similar to the format of the reports menu model. Refer to the section, Reports Menu Model XML Items.
6. Test the Retail Application.

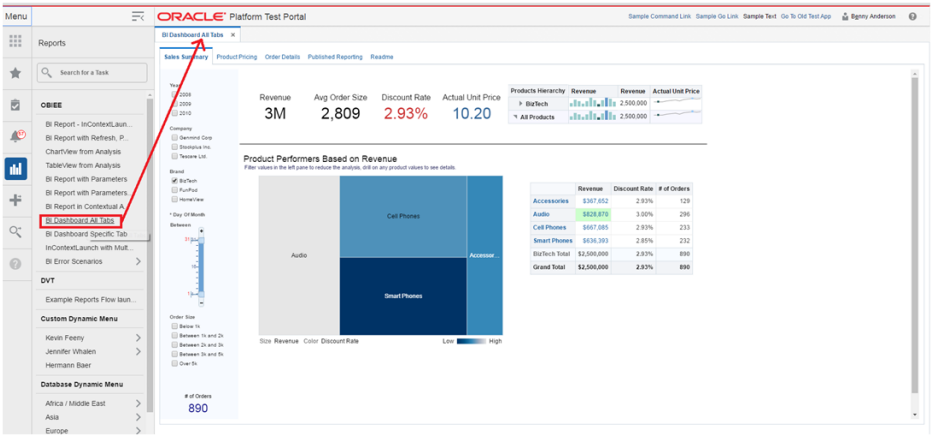
Dashboard Customization Scenarios

Understanding Dashboards in Retail Applications

A dashboard is a page within a Retail Application that displays the status of metrics and key performance indicators relevant to the Retail Application. They are typically tailored for specific roles and they allow users to easily monitor the status of the current data within the application.

Users can access dashboards from the application's Reports menu.

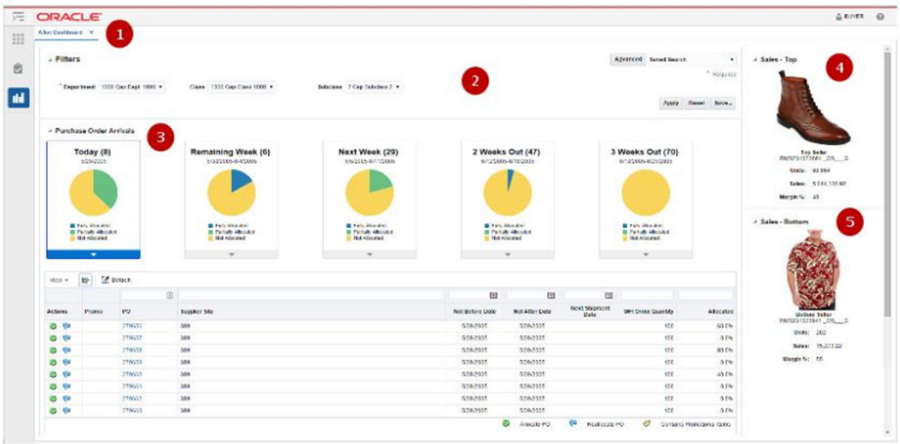
The example below shows the dashboard page as rendered in the local area of a Retail Application's UI.



Anatomy of a Dashboard

Depicted below is an example of a dashboard that might appear in a Retail Application:

Figure 7–5 Dashboard Example



1	Dashboard Page	A page rendering a dashboard. A dashboard is composed of 1 or more reports. The dashboard page is launched as a tab within the Retail Application

2	Prompt Region	The prompt region contains a set of editable input components that allow users to modify the parameters of the various reports in the dashboard. This allows users to see different sets of data for the same set of reports. This is optional on a dashboard page. There can be multiple prompt regions in a dashboard affecting different sets of reports on a dashboard.
3, 4, 5	Report Regions	A report region shows a graphical view of the application's business data.

Supported Implementation of Dashboards

Retail Applications support the following implementation of dashboards:

- Dashboards built using Oracle ADF.
- Dashboards built in external BI reporting tools and accessible via a web browser URL.
- Dashboards built using Oracle ADF is the recommended implementation of dashboards as they have seamless integration with the rest of the Retail Application.
- Dashboards built in external BI reporting tools are supported as long as the dashboards can be accessed in a web browser via a URL. Oracle Retail recommends using Oracle Business Intelligence Enterprise Edition (OBIEE) for URL based dashboards.
- Dashboards developed externally doesn't support Contextual links to Allocation Application, that means report can not access the App using contextual links.

Retail Application Included Dashboards

As mentioned in the section, Understanding the Deployment of Retail Applications, a Retail Application is installed with a shared library containing included dashboards that are rendered by default within the application.

These dashboards have the following characteristics:

- The dashboards are implemented in Oracle ADF for seamless integration with the rest of the application workflows.
- The dashboards are implemented using a common framework called ADF Dashboard Prompts to allow for a standard way of customizing them.

Retailers can customize included dashboards. Please refer to the section, Dashboard Customization Scenarios for details on the types of customization retailers can perform on included dashboards.

Adding a New ADF-based Dashboard in the Reports Menu

This section describes the steps for adding ADF based dashboard reports into the Retail Application's UI Sidebar. ADF-based reports can be integrated into the Retail Application by adding the dashboard report's task flow URL in the application's reports menu model XML file:

1. Add a new custom shared library. Refer to the section, Adding a Custom Shared Library.
2. Implement the dashboard page components in the custom shared library workspace. Dashboard page components are built as Oracle ADF bounded task

flows. Refer to the section, Creating New ADF Contents, for important considerations when building custom ADF content.

3. Note the task flow URL and the parameters of the dashboard task flow.
4. Register the task flow URL and parameters into the application's Reports Menu Model. Refer to section, Adding or Modifying an Item in the Reports Menu.
5. Re-generate the Custom Shared Library WAR file from the Custom Shared Library workspace. Shutdown the Retail Application and its Custom Shared Library Registry, redeploy the Custom Shared Library WAR file, and restart the Retail Application components.
6. Grant security permissions to dashboard page components. Refer to the section, Creating New ADF Contents.
7. Test the Retail Application. Log-in and verify the newly added content.

Adding a New External Dashboard into the Reports Menu

This section describes the steps for adding external dashboard reports into the Retail Application's UI Sidebar. The reports such as OBIEE, BI Publisher or Microstrategy can be integrated into the Retail Application by adding the web URL of the dashboard report in the application's reports menu model XML file:

1. Create the dashboard report in the reporting server instance.
2. Obtain the URL to the dashboard report along with any parameters needed to render it.
3. Add a New Custom Shared Library. Refer to the section, Adding a Custom Shared Library.
4. Open the Custom Shared Library workspace in JDeveloper.
5. Add a new item in the reports menu for the URL and parameters of the dashboard report. Refer to the section, Adding or Modifying an Item in the Reports Menu.
6. Re-generate the Custom Shared Library WAR file from the Custom Shared Library workspace. Shutdown the Retail Application and its Custom Shared Library Registry, redeploy the Custom Shared Library WAR file, and restart the Retail Application components.
7. Test the Retail Application. Log-in and verify that a link to the BI dashboard appears in the UI's sidebar task tree under a folder "My Custom Dashboards".

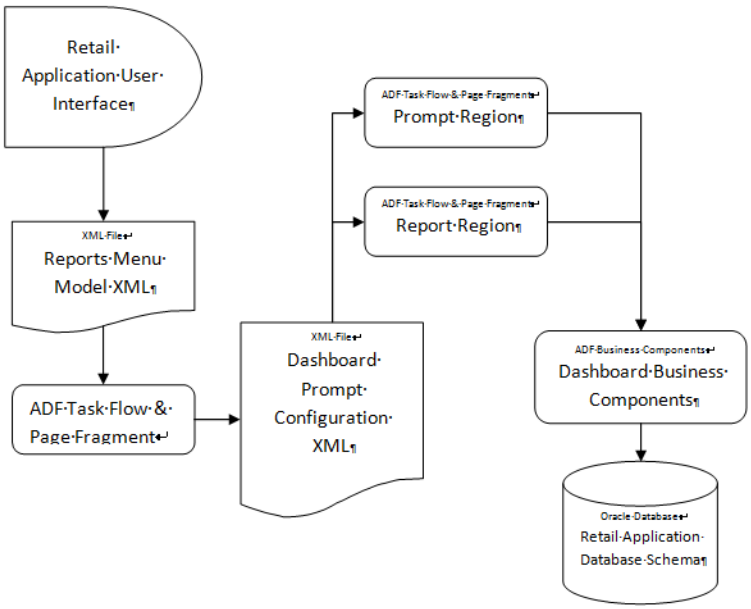
Retail Application Included Dashboard Customization Scenarios

This section contains supported customization use cases for out-of-the-box dashboards included in the Retail Application installation.

Understanding Design Pattern of Included Dashboards

As mentioned in the section, Anatomy of a Dashboard, dashboards are composed of prompt regions and report regions.

The representation of those parts at runtime are depicted in the diagram below:

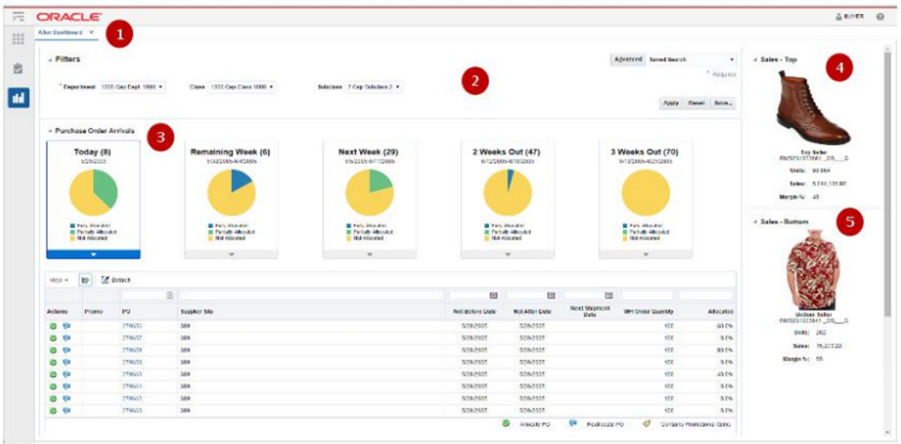


The dashboard page, prompts and reports are implemented as ADF bounded task Flows and page fragments. The prompts and regions are configured in an XML file called a Dashboard Prompt Configuration XML. The dashboard prompt XML arranges visually the prompts and regions on the dashboard page.

The dashboard task flow is launchable from the Retail Application User Interface's Reports Menu. In order to make the report launchable, the dashboard task flow is registered in the application's reports menu model XML.

A set of ADF business components enable data from the Retail Application schema to be rendered in the report regions.

The figure below shows a rendering of a dashboard (called Alloc Dashboard) in a Retail Application:



The different parts of this example dashboard are described below:

1	Dashboard Page	For this example - this called the Allocations Dashboard or Alloc Dashboard. It contains 1 prompt and 3 reports. It is implemented as an ADF bounded task flow called AllocDashboardFlow.
2	Prompt Region	This is the prompt region that users modify to change the view of the various reports in this dashboard. It is implemented as an ADF bounded task flow called DashboardFilterFlow.
3	Order Status Report	A report region showing a row of information tiles and a detail table. It is implemented as an ADF bounded task flow called OrderStatusFlow.
4	Top Seller Report	A report region showing the top selling item. It is implemented as an ADF bounded task flow called TopSellerFlow
5	Bottom Seller Report	A report region showing the bottom selling item. It is implemented as an ADF bounded task flow called BottomSellerFlow

The dashboard is launched from the application's reports menu. The reports menu model XML file (as shown below) contains an item called allocDashboard registering the task flow, AllocDashboardFlow, into the reports menu. Note that the dashboard task flow includes a parameter called dashboardConfigXML which references the location of the Dashboard Prompt Configuration XML file.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<NavigationDefinition id="Folder_1"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="urn:www.oracle.com:oracle.retail.apps.framework.navigation"
    xsi:schemaLocation="urn:www.oracle.com:oracle.retail.apps.framework.navigation
classpath:oracle/retail/apps/framework/uishell/navigation/model/schema/NavigationM
odel.xsd">
    <Items>
        <Item id="allocDashboard" title="Alloc Dashboard"
              shortDesc="Alloc Dashboard"
              type="taskflow">
            <url>/WEB-INF/AllocDashboardFlow.xml#AllocDashboardFlow</url>
            <Parameters>
                <Parameter
id="dashboardConfigXML">"/oracle/AllocDashboardPrompt.xml"</Parameter>
            <Parameters>
        </Item>
    </Items>
</NavigationDefinition>
```

The Dashboard Prompt Configuration XML file for this example is called the AllocDashboardPrompt.xml and it contains the following entries:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Dashboard layout="column"
xmlns="urn:www.oracle.com:oracle.retail.apps.framework.dashboard"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
xsi:schemaLocation="urn:www.oracle.com:oracle.retail.apps.framework.dashboard
classpath:oracle/retail/apps/framework/dashboard/model/schema/dashboardSchema.xsd"
>
    <Vectors>
        <Vector>
            <Items>
                <Item id="filter" type="prompt">

<url>/WEB-INF/DashboardFilterFlow.xml#DashboardFilterFlow</url>
                </Item>
                <Item id="orderStatus" type="report">

<url>/WEB-INF/oracle/retail/apps/framework/dvtcontextawarereport/publicui/flow/DVT
ContextAwareReportFlow.xml#DVTContextAwareReportFlow</url>
                    <Parameters>
                        <Parameter
id="taskflowURL">/WEB-INF/OrderStatusFlow.xml#OrderStatusFlow</Parameter>
                        <Parameter id="parameterName1">departmentIds</Parameter>
                        <Parameter id="payloadKeyName1">DepartmentId</Parameter>
                        <Parameter id="parameterName2">classIds</Parameter>
                        <Parameter id="payloadKeyName2">ClassId</Parameter>
                        <Parameter id="parameterName3">subclassIds</Parameter>
                        <Parameter id="payloadKeyName3">SubclassId</Parameter>
                    </Parameters>
                </Item>
            </Items>
        </Vector>
        <Vector width="300px">
            <Items>
                <Item id="topSeller" type="report">

<url>/WEB-INF/oracle/retail/apps/framework/dvtcontextawarereport/publicui/flow/DVT
ContextAwareReportFlow.xml#DVTContextAwareReportFlow</url>
                    <Parameters>
                        <Parameter
id="taskflowURL">/WEB-INF/oracle/retail/apps/flow/TopSellerFlow.xml#TopSellerFlow<
/Parameter>
                        <Parameter id="parameterName1">departmentIds</Parameter>
                        <Parameter id="payloadKeyName1">DepartmentId</Parameter>
                        <Parameter id="parameterName2">classIds</Parameter>
                        <Parameter id="payloadKeyName2">ClassId</Parameter>
                        <Parameter id="parameterName3">subclassIds</Parameter>
                        <Parameter id="payloadKeyName3">SubclassId</Parameter>
                    </Parameters>
                </Item>
                <Item id="bottomSeller" type="report">

<url>/WEB-INF/oracle/retail/apps/framework/dvtcontextawarereport/publicui/flow/DVT
ContextAwareReportFlow.xml#DVTContextAwareReportFlow</url>
                    <Parameters>
                        <Parameter
id="taskflowURL">/WEB-INF/oracle/retail/apps/flow/BottomSellerFlow.xml#BottomSelle
rFlow</Parameter>
                        <Parameter id="parameterName1">departmentIds</Parameter>
                        <Parameter id="payloadKeyName1">DepartmentId</Parameter>
                        <Parameter id="parameterName2">classIds</Parameter>
                        <Parameter id="payloadKeyName2">ClassId</Parameter>
                        <Parameter id="parameterName3">subclassIds</Parameter>
                        <Parameter id="payloadKeyName3">SubclassId</Parameter>
```

```
                    </Parameters>
                </Item>
            </Items>
        </Vector>
    </Vectors>
</Dashboard>
```

The file registers the prompts and regions into a hierarchy of vectors and items. The structure of the dashboard prompt configuration XML file is explained in the following section.

The Dashboard Prompt Configuration XML File

The root element of the model is the "Dashboard" element. Other than schema and namespace information, the "Dashboard" element has a "layout" attribute that must be provided, and must contain a value of either "row" or "column". This attribute arranges the regions of the dashboard in rows or columns.

Inside the "Dashboard" element is a "Vectors" element, which contains a list of "Vector" elements. If the dashboard layout="row", then each "Vector" corresponds to a single row in the dashboard. Conversely, if layout="column", then each "Vector" corresponds to a column. To add a new row or column, simply add another "Vector" element inside the "Vectors" element.

Up to 12 rows and 6 columns can be displayed in a dashboard, not including separate header and footer regions. Any items beyond 12 rows or 6 columns will not appear on the dashboard.

Rows in the dashboard display from top to bottom. If layout="row", the first "Vector" will appear toward the top of the page. Columns appear from left to right, so if layout="column", the first "Vector" will appear toward the left of the page.

Each "Vector" element contains an "Items" element, which contains a list of "Item" elements. The xml structure for "Item" elements is very similar to the tasks or reports sidebar menu models.

The following attributes are supported on dashboard items:

Name	Description	Required?
id	A unique id for this region	Yes
type	The type of content contained in this region. Valid values: prompt, report	Yes
width	The width of this region. See below for more details.	No
height	The height of a region. See below for more details.	No
url	The task flow url to open in this region	Yes
Parameters	Parameters to be sent to the URL	No

In addition to the items arranged in rows or columns, you can add a header and footer region to the dashboard. Both horizontally span the full width of the dashboard, with the header appearing at the top and the footer appearing at the bottom. These can be added in the model by specifying a "HeaderItem" or "FooterItem" child of the root "Dashboard" element. These regions support all the same properties as other items in the dashboard.

Vectors and items expose width and height attributes to enable applications to resize elements to best fit on the dashboard. Certain rules and best practices apply when using these attributes.

First, if the dashboard is in row layout, then all items in a row must have the same height. The height attribute from the vector is used to specify the height of every item in that row. Since rows are assumed to span the full width, the vector width is ignored in this layout.

In the same way, when the dashboard is arranged in column layout, all items in a column must have the same width. The vector width is used to specify the width of every item in that column. Columns should expand vertically to accommodate all content, allowing a scrollbar if necessary. Therefore, vector height is ignored in this layout.

In addition to the width and height attributes, a vector also provides a dimensionsFrom attribute. The valid values of the dimensionsFrom attribute can be auto, child or parent. The dimensionsFrom attribute is used to specify whether the Items in the vector inherit their dimensions from the parent decorative box or not. When the dimensionsFrom attribute is parent, the Items in the vector will be stretched to fill any available space in the vector. When the dimensionsFrom attribute is child, the Items in the vector won't stretch but will display scroll bars if they cannot be accommodated in the available space in the decorative box. The default value of the dimensionsFrom attribute is 'auto' which gives preference to the child dimensions.

The following table indicates which attributes are used, based on which dashboard layout has been selected.

	Row layout	**Column layout**
Vector width	ignored	valid
Vector height	valid	ignored
Item height	ignored	valid
Item width	valid	ignored

In column layout, either every vector width should be provided, or every width should be omitted. If all widths are omitted, columns default to divide the total width evenly. When the widths are omitted, the dimensionsFrom attribute of the vector can be set to parent if stretching of the Items is desired. If widths are provided, they should be specified as a percentage, and the sum of all widths should equal 100%.

In row layout, heights may be specified or omitted as desired. If a row's height is omitted, it will default to get the dimension from the content inside the regions. When the row's height is omitted, the dimensionsFrom attribute of the vector can be set to parent if stretching of the Items is desired. Heights should be specified using an exact unit such as px or em. Percentages should not be used.

If these rules and best practices are ignored, results may be inconsistent or undesirable. Items may not size as intended, and horizontal scrollbars or nested vertical scrollbars may appear.

In addition to resizing entire rows or columns, individual items inside the row or column can also be resized. As mentioned above, items in a row should all have the same height, so in row layout, item height is ignored and only item width is valid. And items in a column should have the same width, so item width is ignored and only item height is valid. See the table above for details.

The same rules and best practices that apply to vector height and width also apply to setting item height and width.

Refreshing Reports on Prompt Changes

When the user changes a value on the dashboard prompt, that value translates as a parameter change on the different reports on the dashboard.

This is accomplished through a feature in ADF called Contextual Events. To put simply, contextual events are signals that a page in a task flow can generate for which pages in other task flows can listen and react to. For the included dashboards, prompt regions generate a contextual event whenever the user changes a prompt field value. When a report "hears" an event, it will process the contextual event information (called a payload) to extract the changed prompt value, use that value as a parameter to the report query and refresh the report region.

In the example depicted in the previous section, the dashboard prompt configuration XML file shows all three reports registered under a wrapper task flow called the DVTContextAwareReportFlow.

```
                <Item id="bottomSeller" type="report">

<url>/WEB-INF/oracle/retail/apps/framework/dvtcontextawarereport/publicui/flow/DVT
ContextAwareReportFlow.xml#DVTContextAwareReportFlow</url>
                    <Parameters>
                        <Parameter
id="taskflowURL">/WEB-INF/oracle/retail/apps/flow/BottomSellerFlow.xml#BottomSelle
rFlow</Parameter>
                        <Parameter id="parameterName1">departmentIds</Parameter>
                        <Parameter id="payloadKeyName1">DepartmentId</Parameter>
                        <Parameter id="parameterName2">classIds</Parameter>
                        <Parameter id="payloadKeyName2">ClassId</Parameter>
                        <Parameter id="parameterName3">subclassIds</Parameter>
                        <Parameter id="payloadKeyName3">SubclassId</Parameter>
                    </Parameters>
                </Item>
```

The DVTContextAwareReportFlow wrapper task flow enables a report to listen to contextual events generated by prompt regions.

Note the pairs of parameterNameN and payloadKeyNameN registered in the <Parameters> section for the report task flow BottomSellerFlow. A pair of parameter name and payload key name tells the framework what prompt value is mapped to a specific report parameter. From the example above, parameterName1 is specified as "departmentIds" and payloadKeyName1 as "departmentIds". This means the following:

- There is a task flow parameter for the BottomSellerFlow called "departmentIds"
- When the user changes the value of the department ID in the prompt region, the system will generate a contextual event with the new department value stored as reference "DepartmentId".
- The report is context aware (as indicated by the wrapper task flow - DVTContextAwareReportFlow). When the report detects the contextual event, the framework will get the new department ID value from the payload by using the key "DepartmentId", use it to update the BottomSellerFlow task flow's "departmentId" parameter, and refresh the report region.

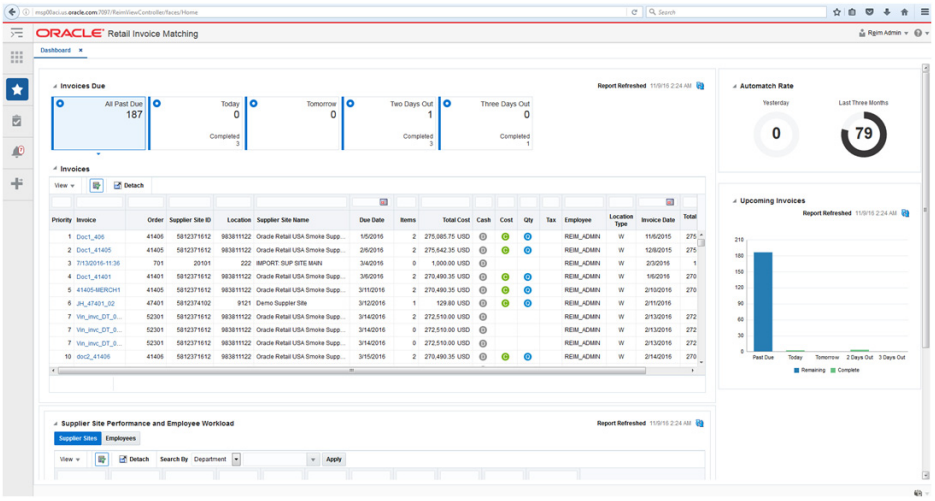
For more details on the DVTContextAwareReportFlow, refer to the section Adding a DVT Taskflow-based Contextual Report.

List of Retail Application Included Dashboards

This section contains a listing of all dashboard reports included in the Retail Application.

- Accounts Payable Summary View
- Finance Manager Supplier View

Accounts Payable Summary View



The dashboard surfaces information in a manner that will help users (Accounts Payable specialist) understand invoice match workload and provide quick visibility into cost, quantity and tax discrepancies. There are 3 main sections of the Accounts Payable dashboard: Upcoming Invoices, Auto Match Rate, and Invoice list. This view will show details for only the invoices that this role is responsible for managing and manually matching.

The dashboard can be accessed by the following job roles:

Accounts Payable Specialist

The following reports will be displayed on the dashboard:

- Upcoming Invoices
- Auto Match Rate
- Invoice list

Task Flow Urls:

/WEB-INF/oracle/retail/apps/reim/dashboards/view/reimdashboard/flow/ReimDashboardFlow.xml

/WEB-INF/oracle/retail/apps/reim/dashboards/view/reimdashboard/flow/UpcomingInvoicesReportFlow.xml

/WEB-INF/oracle/retail/apps/reim/dashboards/view/reimdashboard/flow/AutomatchRateReportFlow.xml

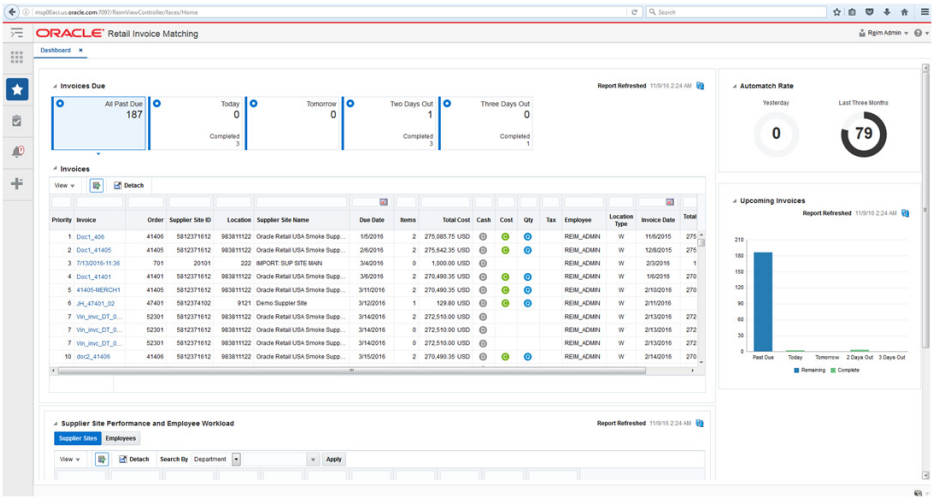
/WEB-INF/oracle/retail/apps/reim/dashboards/view/reimdashboard/flow/InvoiceReportFlow.xml

Parameters to the dashboard task flow:

Parameter	Sample Value
dashboardXml	/oracle/retail/apps/reim/dashboards/view/reimdashboard/ReimDashboard.xml

There are no dashboard prompts for this dashboard

Finance Manager Supplier View



The dashboard surfaces information in a manner that will help users (Finance Manager) understand invoice match workload, view supplier performance, manage employee workload and provide quick visibility into cost, quantity and tax discrepancies. There are 4 main sections for the Finance Manager view; Upcoming Invoices, Auto Match Rate, Suppliers and Invoices. Finance Manager will see the detail for all of his/her employees that he/she manages.

The dashboard can be accessed by the following job roles:

Finance Manager - A Finance Manager is responsible for the financial accounting of a region.

The following reports will be displayed on the dashboard:

- Upcoming Invoices
- Auto Match Rate
- Supplier Performance and Employee Workload
- Invoices

Task Flow Urls:

/WEB-INF/oracle/retail/apps/reim/dashboards/view/reimdashboard/flow/ReimDashboardFlow.xml

/WEB-INF/oracle/retail/apps/reim/dashboards/view/reimdashboard/flow/UpcomingInvoicesReportFlow.xml

/WEB-INF/oracle/retail/apps/reim/dashboards/view/reimdashboard/flow/AutomatchRateReportFlow.xml

/WEB-INF/oracle/retail/apps/reim/dashboards/view/reimdashboard/flow/SupplierSitePerformanceReportFlow.xml

/WEB-INF/oracle/retail/apps/reim/dashboards/view/reimdashboard/flow/EmployeeWorkloadReportFlow.xml

/WEB-INF/oracle/retail/apps/reim/dashboards/view/reimdashboard/flow/InvoiceReportFlow.xml

Adding or Replacing A Report In An Included Dashboard

This section discusses the steps needed to add or replace a report on an included dashboard.

1. Add a custom shared library. Refer to the section, Adding a Custom Shared Library. An existing custom shared library can be reused.
2. Identify the dashboard to be modified. Refer to the section, List of Retail Application Included Dashboards.
3. If needed, open the custom shared library workspace in JDeveloper and build your own custom report as an ADF bounded task flow. Refer to the section, Creating New ADF Contents.

   If the report has to refresh when a prompt on the dashboard is changed by the user, make sure that task flow parameters are present for the prompt values your report will need to react to. Refer to the section, List of Retail Application Included Dashboards, for a list of prompt payload values that can be generated for the dashboard.
4. Note the new report's task flow URL and parameters.
5. Obtain a copy of the dashboard prompt configuration XML file for the dashboard to be modified. Refer to the section, List of Retail Application Included Dashboards.
6. Modify the copy of the dashboard prompt configuration XML file to add an item or replace an existing item for the new report. Refer to the section, The Dashboard Prompt Configuration XML File for details on properly creating a new item entry in the configuration file.

   If the report has to refresh when a prompt on the dashboard is changed by the user, make sure that report task flow is wrapped in a DVTContextAwareTaskFlow for its item entry in the dashboard prompt configuration XML file. Use the example in the section, Understanding Design Pattern of Included Dashboards, as reference. Refer to the section, Adding a DVT Taskflow-based Contextual Report for usage of the DVTContextAwareTaskFlow framework.
7. Modify the included dashboard's entry in the application's Reports Menu to reference the location of your copy of the dashboard prompt configuration XML file. Refer to the section, Adding or Modifying an Item in the Reports Menu.
8. Re-generate the Custom Shared Library WAR file from the Custom Shared Library workspace. Shutdown the Retail Application and its Custom Shared Library Registry, redeploy the Custom Shared Library WAR file, and restart the Retail Application components.
9. Test the Retail Application.

Removing a Report From An Included Dashboard

This section discusses the steps needed to remove a report from an included dashboard.

1. Add a custom shared library. Refer to the section, Adding a Custom Shared Library. An existing custom shared library can be reused.

2. Identify the dashboard to be modified. Refer to the section, List of Retail Application Included Dashboards.
3. Obtain a copy of the dashboard prompt configuration XML file for the dashboard to be modified. Refer to the section, List of Retail Application Included Dashboards.
4. Modify the copy of the dashboard prompt configuration XML file and remove the report. You may need to readjust the position of the other reports. Refer to the section, The The Dashboard Prompt Configuration XML File for details on the configuration XML attributes that control the rendering of the dashboard.
5. Modify the included dashboard's entry in the application's Reports Menu to reference the location of your copy of the dashboard prompt configuration XML file. Refer to the section, Adding or Modifying an Item in the Reports Menu.
6. Re-generate the Custom Shared Library WAR file from the Custom Shared Library workspace. Shutdown the Retail Application and its Custom Shared Library Registry, redeploy the Custom Shared Library WAR file, and restart the Retail Application components.
7. Test the Retail Application.

Change the Layout Of An Included Dashboard

This section discusses the steps needed to change the layout of the reports in included dashboards.

1. Add a custom shared library. Refer to the section, Adding a Custom Shared Library. An existing custom shared library can be reused.
2. Identify the dashboard to be modified. Refer to the section, List of Retail Application Included Dashboards.
3. Obtain a copy of the dashboard prompt configuration XML file for the dashboard to be modified. Refer to the section, List of Retail Application Included Dashboards.
4. Modify the copy of the dashboard prompt configuration XML file and rearrange the position of the item entries. Refer to the section, The Dashboard Prompt Configuration XML File for details on the attributes to control the layout of dashboard.
5. Modify the included dashboard's entry in the application's Reports Menu to reference the location of your copy of the dashboard prompt configuration XML file. Refer to the section, Adding or Modifying an Item in the Reports Menu.
6. Re-generate the Custom Shared Library WAR file from the Custom Shared Library workspace. Shutdown the Retail Application and its Custom Shared Library Registry, redeploy the Custom Shared Library WAR file, and restart the Retail Application components.
7. Test the Retail Application.

Adding Contextual Reports

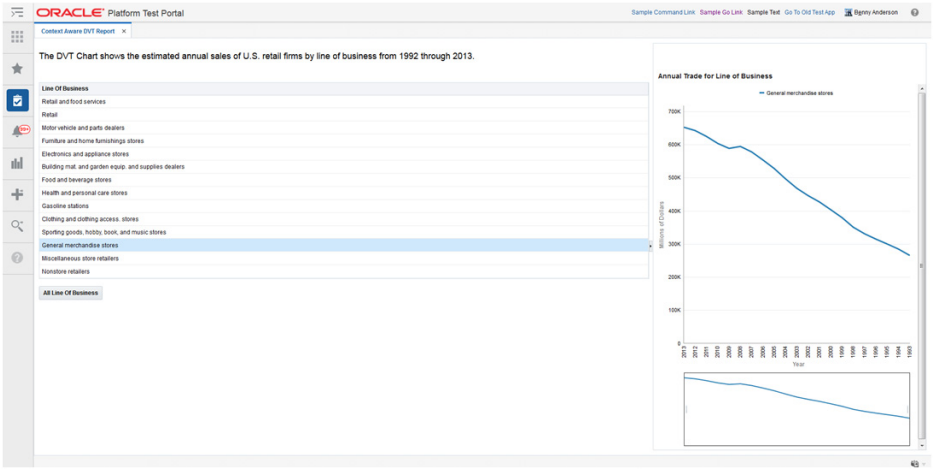
Contextual Reports are reports that appear in a task flow's contextual area section. As discussed in the section, Understanding the Retail Application User Interface, the Contextual Area is a collapsible section to the right of the local area that provides a space to present information that can assist users in completing their tasks.

Since information presented in a contextual area is presented depending on the task or workflow the user is on, contextual areas are associated to task flows and there can be at most one contextual area per task flow.

Within a contextual area - multiple contextual reports can be configured.

Each contextual report can change its contents depending on the action being performed in the user's current workflow.

For example, in the screenshot below, as the user selects the line of business from the table, the line chart will show the annual trade for the selected line of business.



Each task flow publishes contextual business events on key activities happening in the screen. Contextual reports can listen to those events and change its content depending on the payload information associated with the event.

Contextual reports can either be ADF based DVT reports or can be maintained in a separate BI reporting tool. The ADF based DVT reports are ADF taskflows, which are built using the ADF Data Visualization components. The reports maintained in an external reporting tool are URLs accessible in a web browser via a URL. Oracle Retail recommends using Oracle Business Intelligence Enterprise Edition (Oracle BI EE) for an external reporting tool.

The contextual reports are added into a Retail Application by adding and configuring the task flow's contextual area model XML file into the Custom Shared Library.

Retail Applications may provide retailers with a list of possible contextual business events each flow generates. Retailers can configure their contextual reports to react to these events. Each event will also include information about the event's payload information (example: the item ID of the item being selected in an Allocation Maintenance screen).

List of Contextual Business Events and Payloads

Event Name	Task Flow	Contextual Area Name	Contextual Area Model XML Location	Generated when…	Payload Values
DetailToleranceRange	ToleranceRangeDetailFlow	ToleranceRangeDetail	ReimViewController.EAR ' ReimDashboardSharedLib.war ' /oracle/retail/apps/reim/dashboards/view/uishell/config/custom/contextualarea/ContextualAreaModelDetailMatchFlow.xml	The user enters the Detail Match screen or clicks on an item from the table	▪ sessionId - the session id ▪ workspaceId ▪ itemViewId - the item view id ▪ invcItemViewId the invoice item view id ▪ rowType - Either STYLE / SKU / INVC / RCPT to indicate the row type the cursor is currently on ▪ stagedPayloadId a unique payload value generated by the Detail Match page to trigger the contextual report even when the payload do not change ▪

Event Name	Task Flow	Contextual Area Name	Contextual Area Model XML Location	Generated when…	Payload Values
qtyComparison	QuantityComparisonFlow	QuantityComparison	ReimViewController.EAR ' ReimDashboardSharedLib.war ' /oracle/retail/apps/reim/dashboards/view/uishell/config/custom/contextualarea/ContextualAreaModelDetailMatchFlow.xml	The user enters the Detail Match screen or clicks on an item from the table	■ sessionId - the session id ■ workspaceId ■ itemViewId - the item view id ■ invcItemViewId the invoice item view id ■ rowType - Either STYLE / SKU / INVC / RCPT to indicate the row type the cursor is currently on ■ stagedPayloadId a unique payload value generated by the Detail Match page to trigger the contextual report even when the payload do not change ■

Event Name	Task Flow	Contextual Area Name	Contextual Area Model XML Location	Generated when…	Payload Values
costComparison	CostEventFlow	CostEvent	ReimViewController.EAR ' ReimDashboardSharedLib.war ' /oracle/retail/apps/reim/dashboards/view/uishell/config/custom/contextualarea/ContextualAreaModelDetailMatchFlow.xml	The user enters the Detail Match screen or clicks on an item from the table	▪ sessionId - the session id ▪ workspaceId ▪ itemViewId - the item view id ▪ invcItemViewId the invoice item view id ▪ rowType - Either STYLE / SKU / INVC / RCPT to indicate the row type the cursor is currently on ▪ stagedPayloadId a unique payload value generated by the Detail Match page to trigger the contextual report even when the payload do not change ▪
suppSiteEventBinding	SupplierSiteFlow	SupplierSite	ReimViewController.EAR ' ReimDashboardSharedLib.war ' /oracle/retail/apps/reim/dashboards/view/uishell/config/custom/contextualarea/ContextualAreaModelDiscrepancyListFlow.xml	The user searches from the Discrepancy List screen or clicks on an item from the table	siteId - the supplier site id
qtyEventBinding	QuantityComparisonFlow	QuantityComparison	ReimViewController.EAR ' ReimDashboardSharedLib.war ' /oracle/retail/apps/reim/dashboards/view/uishell/config/custom/contextualarea/ContextualAreaModelDiscrepancyListFlow.xml	The user searches from the Discrepancy List screen or clicks on an item from the table	▪ sessionId - the session id ▪ workspaceId ▪ discViewId - the discrepancy list view id

Event Name	Task Flow	Contextual Area Name	Contextual Area Model XML Location	Generated when…	Payload Values
costEventBinding	CostEventFlow	CostEvent	ReimViewController.EAR ' ReimDashboardSharedLib.war ' /oracle/retail/apps/reim/dashboards/view/uishell/config/custom/contextualarea/ContextualAreaModelDiscrepancyListFlow.xml	The user searches from the Discrepancy List screen or clicks on an item from the table	▪ sessionId - the session id ▪ workspaceId ▪ discViewId - the discrepancy list view id
SummaryToleranceRange	ToleranceRangeSummaryFlow	ToleranceRangeSummary	ReimViewController.EAR ' ReimDashboardSharedLib.war ' /oracle/retail/apps/reim/dashboards/view/uishell/config/custom/contextualarea/ContextualAreaModelSummaryMatchFlow.xml	The user enters the Summary Match screen or clicks on an item from the table	▪ sessionId - the session id ▪ workspaceId ▪ stagedPayloadId a unique payload value generated by the Detail Match page to trigger the contextual report even when the payload do not change
supplierSiteEvent	SupplierSiteFlow	SupplierSite	ReimViewController.EAR ' ReimDashboardSharedLib.war ' /oracle/retail/apps/reim/dashboards/view/uishell/config/custom/contextualarea/ContextualAreaModelSummaryMatchFlow.xml	The user enters the Summary Match screen or clicks on an item from the table	▪ siteId - the supplier site id ▪ stagedPayloadId a unique payload value generated by the Detail Match page to trigger the contextual report even when the payload do not change

Preparing the Custom Shared Library for Adding Contextual Reports

This section describes how to prepare the Custom Shared Library so retailers will be able to add contextual reports in Retail Application task flows.

1. Perform the steps described in the section Adding a Custom Shared Library to create a Custom Shared Library workspace, generate a shared library WAR out of it, deploying the WAR, and associating the library to the Retail Application.
2. Using JDeveloper, open the Custom Shared Library workspace in Developer Role.
3. Add new contents as required.
4. Regenerate the shared library WAR file from the workspace and redeploy the shared library. Shutdown and restart of the Retail Application and its shared library registry is required.

5. Changes required in Shared Libarary Registry for Custom Contextual Reports

   a. Obtain a copy of the task flow contextual area model XML files where contextual reports will be added. Refer to section, List of Contextual Business Events and Payloads.

      Example: If the contextual area model XML for the task flow AllocMaintFlow is called AllocMaintFlowContextualAreaModel.xml - then that file's path must be View -Controller/src/custom/AllocMaintFlowContextualAreModel.xml.

   b. Please refer the section Referencing the Custom Shared Library from the Retail Application to register custom contextual reports shared library with in the Retail Applications.

   c. Open the original WAR file using an archive file manager and copy the contextual area model XML file in the View-Controller project src directory, preferably under a subdirectory called custom.

   d. Add a new or open the existing file called PageTemplateOverrideModel.properties under the View-Controller/src directory. Modify this file and add the following entry:

      <ContextualAreaName>.contextualAreaModel=<path to the contextual area model for the flow>
      Example:

      AllocMaintFlow.sidebarModel=/custom/AllocMaintFlowContextualAreaModel.xml

6. Test the Retail Application. Navigate to the flow and make sure the flow is functional.

Adding a URL based Contextual Report

Adding a contextual report to a task flow entails the modification of the task flow's Contextual Area Model XML file. Multiple reports can be added to the model. Each report is rendered in a collapsible panel box.

Before adding a URL based contextual report, the retailer must have:

- Built, prepared, deployed, and tested the Custom Shared Library as described in the section, section Preparing the Custom Shared Library for Adding Contextual Reports.
- Obtained information about the Retail Application's list of contextual business events that can be broadcast from various workflows.
- Created one or more contextual BI reports in the BI reporting tool (e.g. Oracle BI EE).
  - The web URL for each report must be available in order to proceed with the steps in this section.
  - Any parameters to configure the content of the report must be known and should be accessible as parameters to the dashboard's URL.

Once the above pre-requisites have been satisfied, proceed with the following steps:

1. Assume the following example scenario when following the steps:
   - A contextual report called "Item Metrics" showing information about an item is to be added to the Allocation Maintenance Flow's main page. When the user selects an item on the page, the report will show information for the selected item.

- Retail Application has provided the following information about the contextual business event:

Event Name	Task Flow	Page	Contextual Area Model XML Location	Generated when…	Payload Values
AllocMaintItemSelectedEvent	AllocMaintFlow	AllocMaintPage	EAR name -> war name-> jar name -> /oracle/retail/apps/xyz/AllocMaintFlowContextualAreaModel.xml	The user selects an item on the Allocation Maintenance Flow's main screen.	▪ selectedItem - the item selected by the user. Always generated. ▪ selectedItemType - the type of item. Can be "regular" or "pack". Optional. If not supplied, assume item is a regular type.

- The Item Metrics report was built in the retailer's BI reporting tool (e.g. Oracle BI EE). It was built considering the payload values the contextual business event will generate.

2. Open the Custom Shared Library workspace in JDeveloper.
3. Open the task flow contextual area model XML file (ex. ViewController/src/custom/AllocMaintFlowContextualAreaModel.xml).
4. Add an <Item> element within the topmost <Items> element that references the task flow called ViewContextAwareReportFlow. The ViewContextAwareReportFlow is a framework for rendering URL based reports that will be aware of contextual business events emanating from the Retail Application task flows.

   Example:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<NavigationDefinition … >
    <Items>
        ...
        ...

        <Item id="showItemMetric"
              type="taskflow"
              title="Item Metric">
            <url>
/WEB-INF/oracle/retail/apps/framework/contextawarereport/publicui/flow/ViewCont
extAwareReportFlow.xml#ViewContextAwareReportFlow
            </url>
         </Item>

    </Items>
</NavigationDefinition>
```

   Note the following:

- Make sure that the <Item> id is unique.

- Make sure the <Item> type is "taskflow"
- Provide a meaningful title.

5. Populate the parameters to the ViewContextAwareReportFlow by adding the following <Parameter>/<Parameters> elements.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<NavigationDefinition … >
    <Items>

        <Item id="showCustomerMetric"
              type="taskflow"
              title="Customer Metric">
            <url>
/WEB-INF/oracle/retail/apps/framework/contextawarereport/publicui/flow/ViewCont
extAwareReportFlow.xml#ViewContextAwareReportFlow
            </url>

            <Parameters>
                <Parameter id="reportDescription">Item Metric</Parameter>
                <Parameter
id="actionType">AllocMaintItemSelectedEvent</Parameter>
                <Parameter id="primaryUrl">
<![CDATA[/faces/oracle/retail/apps/framework/contextawarereport/publicui/page/V
iewContextAwareReportTestPage.jspx?paramItemId=<selectedItemId>&paramItemType=<
selectedItemType:token01>&paramLanguage=<language>]]>
                </Parameter>
                <Parameter id="token01">regular</Parameter>
             </Parameters>

         </Item>

    </Items>
</NavigationDefinition>
```

Note the following:

- The <Parameter id="reportDescription"> element is the title of the contextual area report. Set this to a meaningful value.
- The <Parameter id="actionType"> element indicates the contextual business event name the report will listen to.
- The <Parameter id="primaryUrl"> element indicates the URL to the contextual area report in the BI server. The entire URL must be marked as character data (e.g. enclosed in CDATA). Note that the parameters to the URL are tokenized. The example above uses a test page called ViewContextAwareReportTestPage.jspx which can be replaced with the actual report URL.
  - The "?paramItemId=<selectedItemId>" portion of the URL instructs the system to pass the contextual business event payload value called selectedItemId into the URL parameter paramItemId when rendering the contextual report.
  - The "?paramItemType=<selectedItemType:token01>" portion of the URL instructs the system to pass the contextual business event payload value called selectedItemType into the URL parameter paramItemType when rendering the contextual report. If that payload value is empty or null at runtime, then a default value of regular is used as referenced in a

<Parameter id="token01"> entry. The colon symbol separates the payload value from the default value if the payload value is null.

  - The "?paramLanguage=<language>" portion of the URL instructs the system about the current locale of the user. The "language" identifier is a reference to a value in the contextual event payload. This is a built-in value that all Retail Application contextual business event payloads will have.

- The <Parameter id="token01"> element holds the default value for the URL parameter selectedItemType. Token parameters hold default values and you can define up to 20 default value tokens.

6. Re-generate the Custom Shared Library WAR file from the Custom Shared Library workspace. Shutdown the Retail Application and its Custom Shared Library Registry, redeploy the Custom Shared Library WAR file, and restart the Retail Application components.
7. Test the Retail Application. Go to the flow where the report was added and verify that the report is rendered correctly.

Adding a DVT Taskflow based Contextual Report

Before adding a DVT taskflow based contextual report, the retailer must have:

- Built, prepared, deployed, and tested the Custom Shared Library as described in the section Preparing the Custom Shared Library for Adding Contextual Reports.
- Obtained information about the Retail Application's list of contextual business events that can be broadcast from various workflows.
- Created one or more taskflows using the ADF DVT components.
  - Any parameters to configure the content of the report must be known and should be accessible as taskflow parameters.

Once the above pre-requisites have been satisfied, proceed with the following steps:

1. Assume the following example scenario when following the steps:
   - A contextual report called "Item Metrics" showing information about an item is to be added to the Allocation Maintenance Flow's main page. When the user selects an item on the page, the report will show information for the selected item.
   - Retail Application has provided the following information about the contextual business event:

Event Name	Task Flow	Page	Contextual Area Model XML Location	Generated when…	Payload Values
AllocMaintItemSelectedEvent	AllocMaintFlow	AllocMaintPage	EAR name -> war name-> jar name -> /oracle/retail/apps/xyz/AllocMaintFlowContextualAreaModel.xml	The user selects an item on the Allocation Maintenance Flow's main screen.	▪ selectedItem - the item selected by the user. Always generated. ▪ selectedItemType - the type of item. Can be "regular" or "pack". Optional. If not supplied, assume item is a regular type.

- The Item Metrics report was built in ADF. It was built considering the payload values the contextual business event will generate.

2. Re-generate the Custom Shared Library WAR file from the Custom Shared Library workspace. Shutdown the Retail Application and its Custom Shared Library Registry, redeploy the Custom Shared Library WAR file, and restart the Retail Application components.
3. Changes Required in Shared Library Registry for Custom DVT Taskflow based Contextual Report
   a. Obtain the task flow contextual area model XML file (ex. ViewController/src/custom/AllocMaintFlowContextualAreaModel.xml).
   b. Add an <Item> element within the topmost <Items> element that references the task flow called DVTContextAwareReportFlow. The DVTContextAwareReportFlow is a framework for rendering ADF DVT based reports that will be aware of contextual business events emanating from the Retail Application task flows.

      Example:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<NavigationDefinition … >
    <Items>
        ...
        ...

        <Item id="showItemMetric"
              type="taskflow"
              title="Item Metric">
            <url>
/WEB-INF/oracle/retail/apps/framework/dvtcontextawarereport/publicui/flow/D
VTContextAwareReportFlow.xml#DVTContextAwareReportFlow
            </url>
         </Item>

     </Items>
</NavigationDefinition>
```

      Note the following:

- Make sure that the <Item> id is unique.

- Make sure the <Item> type is "taskflow"

- Provide a meaningful title.

c. Populate the parameters to the DVTContextAwareReportFlow by adding the following <Parameter>/<Parameters> elements.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<NavigationDefinition … >
    <Items>

        <Item id="showCustomerMetric"
              type="taskflow"
              title="Customer Metric">
            <url>
/WEB-INF/oracle/retail/apps/framework/dvtcontextawarereport/publicui/flow/D
VTContextAwareReportFlow.xml#DVTContextAwareReportFlow
            </url>
            <Parameters>
                <Parameter
id="taskflowURL">/WEB-INF/oracle/retail/apps/framework/uishell/navigation/c
ontextualarea/flow/ItemMetricFlow.xml#ItemMetricFlow</Parameter>
                <Parameter id="parameterName1">itemId</Parameter>
                <Parameter id="payloadKeyName1">ItemId</Parameter>
                <Parameter
id="parameterValue1">#{payload.valueMap['ItemId']}</Parameter>
          <Parameter id="refreshOnDisclosure">#{true}</Parameter>
            </Parameters>
         </Item>

    </Items>
</NavigationDefinition>
```

Note the following:

- The <Parameter id="taskflowURL"> element is the URL of the DVT taskflow that will be displayed in the contextual area.

- The <Parameter id="parameterName1"> element indicates a parameter for the DVT taskflow. A maximum of 15 parameters can be specified.

- The <Parameter id="payloadKeyName1"> element indicates the name of the payload key that maps to the taskflow parameter parameterName1. A maximum of 15 payload keys can be specified. The value of the payload key in the event payload will be used for the parameter in the DVT taskflow.

- The <Parameter id="parameterValue1"> element holds the default value of the DVT taskflow. A maximum of 15 parameter values can be specified.

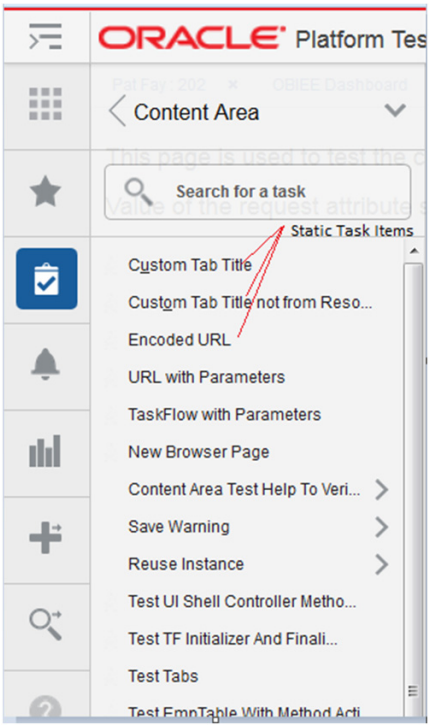
- The <Parameter id="refreshOnDisclosure"> element is used to refresh the DVT taskflow on the disclosure of the tab.

4. Test the Retail Application. Go to the flow where the report was added and verify that the report is rendered correctly.

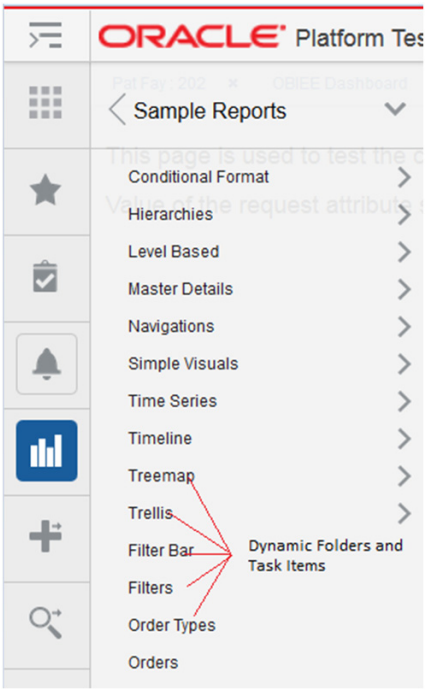
Enabling Dynamic Task Items in the Retail Application

The Retail Application provides the capability to display both static and dynamic tasks in the task list. The static tasks are based off an xml model file, for example,

HomeTaskMenuModel.xml, HomeReportsMenuModel.xml. These tasks should be predefined in the tasks or reports menu model file.



The dynamic tasks can be sourced from any data source like OBIEE, BIPublisher, database tables, Web Services etc. The complete hierarchy of task items including the folders and actions can be generated dynamically at runtime.



From the user's perspective, there is no difference between the static versus the dynamic task items. They look and behave the same.

The DynamicContentHandler Interface

The Retail Application provides an interface called DynamicContentHandler that can be implemented to add dynamic tasks in the task menu. The interface is as follows

```
package oracle.retail.apps.framework.uishell.taskmenu.dynamiccontent.handler;

import java.util.List;

import
oracle.retail.apps.framework.uishell.taskmenu.dynamiccontent.dataobject.TaskMenuIt
em;
import
oracle.retail.apps.framework.uishell.util.RetailUiShellHttpSessionListeners;

public interface DynamicContentHandler extends
RetailUiShellHttpSessionListeners.RemovalListener {
    public List<TaskMenuItem> getChildren(TaskMenuItem taskMenuItem) throws
Exception;

    public List<TaskMenuItem> getDefaultTaskMenuItems() throws Exception;

    public TaskMenuItem getInContextTaskMenuItem(String taskMenuItemId) throws
Exception;
}
```

The interface extends another interface called RetailUiShellHttpSessionListeners.RemovalListener. The RemovalListener interface provides the following method.

```
    public static interface RemovalListener extends Serializable  {
        public void onSessionRemove();
    }
```

The below table explains the use of these methods.

getChildren(TaskMenuItem taskMenuItem)	This method provides the child task items for a given task menu item.
getDefaultTaskMenuItems()	This method provides the task items that will be loaded as the default content in the Retail Application.
getInContextTaskMenuItem(String taskMenuItemId)	This method provides the task item based on a given task menu id for In-Context launching.
onSessionRemove()	This method will be called by the Retail Application when the user logs out of the application. The method can be used to do any resource cleanup or can be used to logout from the external data source.

DynamicContent Type

The Item element of the menu model file supports a type called 'dynamicContent' along with the type's taskflow and link. When the Retail Application encounters the type as dynamicContent, it looks for the attribute called dynamicContentHandler. The dynamicContentHandler attribute will tell the Retail Application the name of the implementation class which will be used to provide the dynamic task items. The class should implement the DynamicContentHandler interface.

```
<Item id="customDynamicMenuFolder" visible="#{true}" title="Custom Dynamic Menu"
              type="folder">
            <Items>
```

```
                <Item id="customTreeModelDynamicContent" title="Custom Dynamic
Menu"
                      type="dynamicContent"

dynamicContentHandler="oracle.retail.apps.framework.uishell.taskmenu.handler.custo
m.CustomTreeModelDynamicContentHandler"
                      visible="#{securityContext.authenticated}"/>
            </Items>
        </Item>
```

The Item element with the type attribute as dynamicContent can be placed anywhere in the hierarchy of the static task items in the tasks or reports menu model file. The dynamic task items will begin exactly from the location where the dynamicContent Item is defined.

The menu model file can be customized to add an entry for the dynamic content as shown above. Please refer to the section Adding or Modifying an Item in the Tasks Menu for more details.

Example Implementation of the DynamicContentHandler Interface

We will start with a simple example to show how to display the dynamic task items in the task menu. The example uses the org.apache.myfaces.trinidad.model.TreeModel implementation called org.apache.myfaces.trinidad.model.RowKeyPropertyTreeModel to pull and display data in the task menu hierarchy. The class org.apache.myfaces.trinidad.model.RowKeyPropertyTreeModel is a subclass of org.apache.myfaces.trinidad.model. ChildPropertyTreeModel. The class ChildPropertyTreeModel creates a TreeModel from a list of beans and the class RowKeyPropertyTreeModel adds the support of row keys to the TreeModel. Let's assume that we have a TreeModel object created for a hierarchy of Employee objects. We want to display the names of the employees as tasks in the task list. The managers will be displayed as folders. The action on the manager's name will open the list of directs for the manager. The action on the leaf employee names will open the employee details screen for the selected employee.

The implementation class for the DynamicContentHandler interface is named as CustomTreeModelDynamicContentHandler. It has a method that builds the TreeModel from the list of Employee objects.

```
/**
 * Build a custom tree model.
 * @return
 */
private TreeModel buildTreeModel() {
    List<Employee> allEmployees = new ArrayList<Employee>();

    Employee manager1 = new Employee(197, "Kevin Feeny");
    Employee emp = new Employee(198, "Donald OConnel");
    manager1.addDirect(emp);
    emp = new Employee(199, "Douglas Grant");
    manager1.addDirect(emp);

    Employee manager2 = new Employee(200, "Jennifer Whalen");
    emp = new Employee(201, "Michael Harstein");
    manager2.addDirect(emp);
    Employee manager3 = new Employee(203, "Susan Marvis");
    emp = new Employee(202, "Pat Fay");
```

```
        manager3.addDirect(emp);
        manager2.addDirect(manager3);

        emp = new Employee(204, "Hermann Baer");

        allEmployees.add(manager1);
        allEmployees.add(manager2);
        allEmployees.add(emp);

        //create a RowKeyPropertyTreeModel with employeeId as a row key and
directs as child property
        return new RowKeyPropertyTreeModel(allEmployees, "directs", "employeeId");
    }
```

The TreeModel is initialized on the initialization of the class CustomTreeModelDynamicContentHandler.

```
    public class CustomTreeModelDynamicContentHandler implements
DynamicContentHandler {

    private static final ADFLogger LOG =
        ADFLogger.createADFLogger(CustomTreeModelDynamicContentHandler.class);
    private TreeModel treeModel;
    private static final String TASKMENU_ITEM_ID_SEPARATOR = "_";

    public CustomTreeModelDynamicContentHandler() {
        super();
        this.treeModel = buildTreeModel();
    }
```

We will implement the getChildren method of the DynamicContentHandler interface.

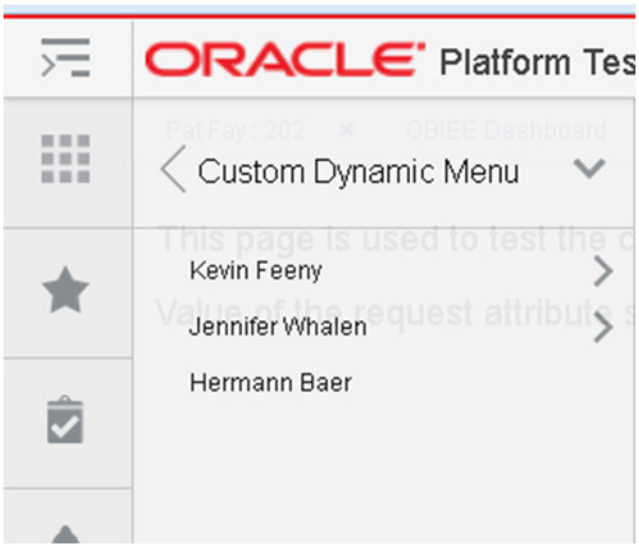
```
    /**
     * This method is called by the Retail Application to get the dynamic task
items.
     * @param taskMenuItem
     * @return
     */
    public List<TaskMenuItem> getChildren(TaskMenuItem taskMenuItem) {
        //if taskMenuItem is null then framework is trying to load the root
objects,
        //otherwise the framework has already navigated the tree
        if (taskMenuItem != null) {
            this.treeModel.setRowKey(taskMenuItem.getPath());
            this.treeModel.enterContainer();
            this.treeModel.setRowIndex(0);
        } else {
            this.treeModel.setRowKey(null); //move to root
        }
        return getChildrenForCurrentRowKey();
    }

    /**
     * This method returns the children for the current row key in the tree model.
     * @param model
     * @return
     */
    private List<TaskMenuItem> getChildrenForCurrentRowKey() {
        List<TaskMenuItem> childTaskMenuItemList = new ArrayList<TaskMenuItem>();
        int rowCount = this.treeModel.getRowCount();
        for (int i = 0; i < rowCount; i++) {
```

```
            this.treeModel.setRowIndex(i);
            TaskMenuItem childTaskMenuItem = buildTaskMenuItem();
            if (childTaskMenuItem != null)
                childTaskMenuItemList.add(childTaskMenuItem);
        }
        return childTaskMenuItemList;
    }
```

The Retail Application will instantiate the class oracle.retail.apps.framework.uishell.taskmenu.handler.custom.CustomTreeModelDynamicContentHandler. It will cache the class for subsequent requests after instantiating it once.

On the initial load of the page, the Retail Application will call getChildern on the dynamicContentHandler implementation class by passing it a null value of the TaskMenuItem object.

When the argument is null, the method getChildern should return the root dynamic task items. The Retail Application will display the root items on the initial load of the page. In the example above, the employees Kevin, Jennifer and Hermann are the root items of the TreeModel so they will be loaded by the Retail Application and displayed as the initial dynamic task items for the CustomTreeModelDynamicContentHandler.



When the user clicks on a Folder, the Retail Application will send the current selected TaskMenuItem object to the getChildern method to load the children of the current selected task item. When the argument of the getChildren method is a valid TaskMenuItem object, the getChildren method should return the children of the TaskMenuItem object.

The Retail Application always lazy loads the dynamic items. This means that the children of a dynamic Folder will only be loaded when the user clicks on the Folder. Once the children are loaded they are cached by the Retail Application. In the above example, even though the Retail Application lazy loads the dynamic items, the TreeModel pre-loads all the employees.

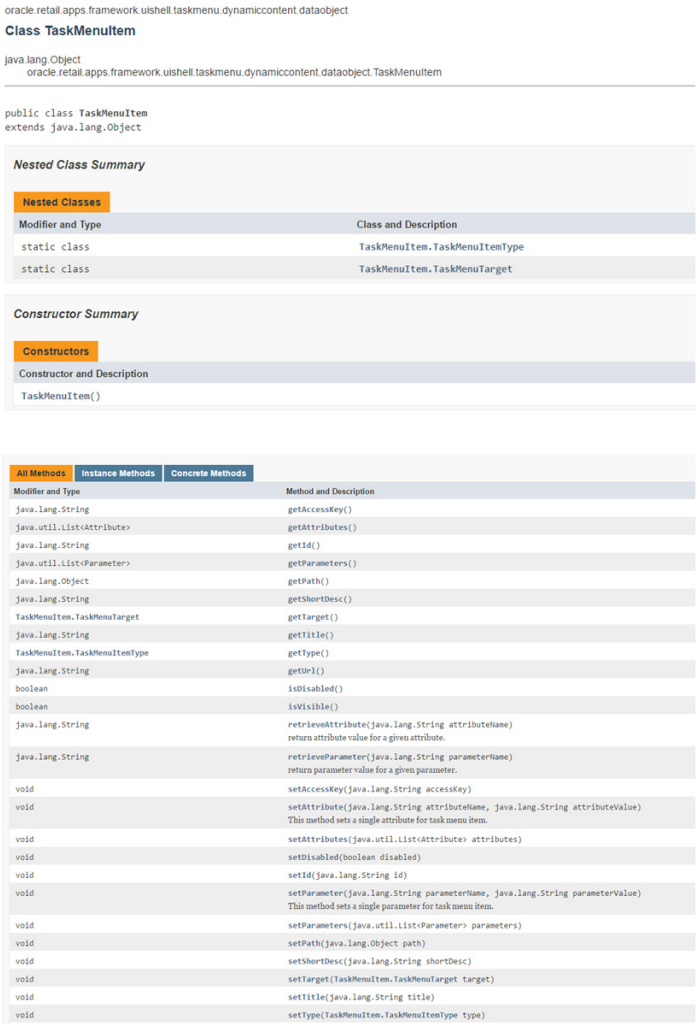
TaskMenuItem class

The TaskMenuItem class is an abstraction of a dynamic task in the task menu. It is an equivalent of the 'Item' element in the task menu model file. The TaskMenuItem class

can be used to add dynamic links, taskflows and folders in the task menu. The class has the following attributes.

```
private Object path;
private String id;
private String title;
private String url;
private String shortDesc;
private boolean visible = true;
private TaskMenuItemType type = TaskMenuItemType.LINK;
private TaskMenuTarget target;
private List<Parameter> parameters;
private List<Attribute> attributes;
private String accessKey;
private boolean disabled;
```

Following is a java doc screenshot of the TaskMenuItem class.



oracle.retail.apps.framework.uishell.taskmenu.dynamiccontent.dataobject

Class TaskMenuItem

java.lang.Object
  oracle.retail.apps.framework.uishell.taskmenu.dynamiccontent.dataobject.TaskMenuItem

public class **TaskMenuItem**
extends java.lang.Object

Nested Class Summary

Nested Classes

Modifier and Type	Class and Description
static class	TaskMenuItem.TaskMenuItemType
static class	TaskMenuItem.TaskMenuTarget

Constructor Summary

Constructors

Constructor and Description
TaskMenuItem()

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method and Description
java.lang.String	getAccessKey()
java.util.List<Attribute>	getAttributes()
java.lang.String	getId()
java.util.List<Parameter>	getParameters()
java.lang.Object	getPath()
java.lang.String	getShortDesc()
TaskMenuItem.TaskMenuTarget	getTarget()
java.lang.String	getTitle()
TaskMenuItem.TaskMenuItemType	getType()
java.lang.String	getUrl()
boolean	isDisabled()
boolean	isVisible()
java.lang.String	retrieveAttribute(java.lang.String attributeName) return attribute value for a given attribute.
java.lang.String	retrieveParameter(java.lang.String parameterName) return parameter value for a given parameter.
void	setAccessKey(java.lang.String accessKey)
void	setAttribute(java.lang.String attributeName, java.lang.String attributeValue) This method sets a single attribute for task menu item.
void	setAttributes(java.util.List<Attribute> attributes)
void	setDisabled(boolean disabled)
void	setId(java.lang.String id)
void	setParameter(java.lang.String parameterName, java.lang.String parameterValue) This method sets a single parameter for task menu item.
void	setParameters(java.util.List<Parameter> parameters)
void	setPath(java.lang.Object path)
void	setShortDesc(java.lang.String shortDesc)
void	setTarget(TaskMenuItem.TaskMenuTarget target)
void	setTitle(java.lang.String title)
void	setType(TaskMenuItem.TaskMenuItemType type)

The attribute 'path' can be used to store the information of the current node. The 'path' attribute is of type 'Object' so it can store any object which can be used to identify the current node in the external hierarchy. In the example above, the path stores the row key of the TreeModel which is an ArrayList of employeeId's starting from the root to the current selected node. The path can vary based on the implementation of the DynamicContentHandler interface. For example, if the DynamicContentHandler

implementation reads the database tables to display a dynamic menu then the path can be an ArrayList of the primary keys of the database tables, if the DynamicContentHandler implementation calls an external web service to generate a dynamic menu then the path can be a String object which identifies a node in the external system.

The following code snippet shows how a TaskMenuItem can be built in the DynamicContentHandler implementation class.

```
    /**
     * build a task menu item from the custom object.
     * @param rowKey
     * @param employee
     * @return
     */
    private TaskMenuItem buildTaskMenuItem() {

        List<Object> rowKey = (List<Object>)this.treeModel.getRowKey();
        Employee employee = (Employee)this.treeModel.getRowData();

        if (this.treeModel.isContainer()) {
            return buildContainerTaskMenuItem(rowKey, employee);
        } else {
            return buildLeafTaskMenuItem(rowKey, employee);
        }
    }

    /**build container taskmenu item.
     * @param rowKey
     * @param employee
     * @return
     */
    private TaskMenuItem buildContainerTaskMenuItem(List<Object> rowKey, Employee
employee) {
        TaskMenuItem taskMenuItem = new TaskMenuItem();
        taskMenuItem.setPath(rowKey);
        taskMenuItem.setId(generateUniqueId(rowKey));
        taskMenuItem.setTitle(employee.getName());
        taskMenuItem.setShortDesc(employee.getName() + " : " +
employee.getEmployeeId());
        taskMenuItem.setType(TaskMenuItem.TaskMenuItemType.FOLDER);
        return taskMenuItem;
    }

    /**build leaf task menu item.
     * @param rowKey
     * @param employee
     * @return
     */
    private TaskMenuItem buildLeafTaskMenuItem(List<Object> rowKey, Employee
employee) {
        TaskMenuItem taskMenuItem = new TaskMenuItem();
        taskMenuItem.setPath(rowKey);
        taskMenuItem.setId(generateUniqueId(rowKey));
        taskMenuItem.setTitle(employee.getName());
        taskMenuItem.setShortDesc(employee.getName() + " : " +
employee.getEmployeeId());
        taskMenuItem.setType(TaskMenuItem.TaskMenuItemType.TASKFLOW);

taskMenuItem.setUrl("/WEB-INF/oracle/retail/apps/framework/uishell/navigation/cont
entarea/flow/TestOverrideCloseTabFlow.xml#TestOverrideCloseTabFlow");
```

```
        //set attributes on taskMenuItem
        taskMenuItem.setAttribute("TabTitle",
                                  employee.getName() + " : " +
employee.getEmployeeId());
        taskMenuItem.setAttribute("tabShortDesc",
                                  "Employee " + employee.getName() + " has emplyee
Id " +
                                  employee.getEmployeeId() + ".");

        //set parameters on taskMenuItem
        taskMenuItem.setParameter("EmployeeId",
String.valueOf(employee.getEmployeeId()));
        return taskMenuItem;
    }

    /**This method generages a unique id for the taskmenu item.
     * @param rowKey
     * @return
     */
    private String generateUniqueId(List<Object> rowKey) {
        return rowKey.stream().map(c ->
c.toString()).collect(Collectors.joining(TASKMENU_ITEM_ID_SEPARATOR));
    }
```

Default Dynamic Task Items

When a user logs in the Retail Application, the user sees the default content that has been configured in the Retail Application. The default content opens up directly in a Retail UIShell tab without the user having to launch it from the task menu. The dynamic task items can also be configured as the default content in the application. The DynamicContentHandler interface provides a method called getDefaultTaskMenuItems which returns a List of TaskMenuItem objects. The items returned from this method will be launched automatically after the user logs in.

The following is a sample implementation of the getDefaultTaskMenuItems method.

```
    /**Get the default task menu items. These will be automatically loaded on
initial load of the home page.
     * @return
     */
    @Override
    public List<TaskMenuItem> getDefaultTaskMenuItems() {
        List<Object> rowKey = new ArrayList<Object>();
        rowKey.add(200); //"Jennifer Whalen"
        rowKey.add(203); //"Susan Marvis"
        rowKey.add(202); //"Pat Fay"
        Employee employee = (Employee)this.treeModel.getRowData(rowKey); //get Pat
Fay

        TaskMenuItem taskMenuItem = buildLeafTaskMenuItem(rowKey, employee);
        List<TaskMenuItem> defaultTaskMenuItems = new ArrayList<TaskMenuItem>();
        defaultTaskMenuItems.add(taskMenuItem);
        return defaultTaskMenuItems;
    }
```

In-Context Launch of Dynamic Task Items

The Retail Application provides a way to launch links and taskflows directly using a browser URL. The content is launched in the context of the external application by passing parameters in the URL.

The Retail Application requires a navModelItemId parameter in the request to identify the content that needs to be launched. The Retail Application checks the static task menu model files to see if an Item exists with that id in the model file. If the Retail Application does not find a static item, it consults the list of dynamic content handlers configured with the application to look for the item. If the dynamic content handler returns a TaskMenuItem for the given id, it will be launched in the Retail Application. The dynamic content handler should implement a method called getInContextTaskMenuItem(String taskMenuItemId) which will return the TaskMenuItem object for the taskMenuItemId received. The getInContextTaskMenuItem should be able to find the Item using the taskMenuItemId.

The following is a sample implementation of the getInContextTaskMenuItem(String taskMenuItemId) method.

```
    /**This method returns an in-context item.
     * @param taskMenuItemId
     * @return
     */
    @Override
    public TaskMenuItem getInContextTaskMenuItem(String taskMenuItemId) {
        TaskMenuItem inContextTaskMenuItem = null;
        Employee employee = null;
        List<Object> rowKey = null;
        if (taskMenuItemId != null) {
            try {
                rowKey =
                    Arrays.stream(taskMenuItemId.split(TASKMENU_ITEM_ID_
SEPARATOR)).map(c -> Integer.valueOf(c)).collect(Collectors.toList());
                employee = (Employee)this.treeModel.getRowData(rowKey);

                if ((employee != null) && (employee.getDirects().isEmpty())) {
                    inContextTaskMenuItem = buildLeafTaskMenuItem(rowKey,
employee);
                }
            } catch (Exception e) {
                if (LOG.isWarning()) {
                    LOG.warning("Error occurred looking for dynamic In-Context
Item with id " +
                                    taskMenuItemId, e);
                }
            }
        }
        return inContextTaskMenuItem;
    }
```

The custom dynamic content handler can be added to the classpath of the Retail Application by deploying it as a shared library. Please refer to the section Adding a Custom Shared Library for details on how to deploy a shared library.

Report Adapters

The DynamicContentHandler interface can be used to add dynamic task items in the task menu from a variety of data sources including OBIEE, BIPublisher, database tables, and custom Web Services etc. The Retail Application provides the default implementation of the DynamicContentHandler interface for OBIEE and BIPublisher. These implementations are called Report Adapters. The Report Adapters can display the user's reports from OBIEE and BIPublisher as tasks in the task list. The user can click on the tasks to display individual reports and dashboards.

Note: The connection to both the OBIEE and BIPublisher should be secured using TLS protocol. Please ensure that the Report Adapter integration is done using secured protocol in the production environments.

OBIEE Report Adapter

The RetailAppsObieeAdapter relies on a properties file called RetailAppsObieeAdapter.properties for the OBIEE configuration. Create RetailAppsObieeAdapter.properties in any location on the file system. Modify the setDomainEnv.cmd for the Windows environment or setDomainEnv.sh file for the Unix environment to add the RetailAppsObieeAdapter.properties as a Java system property specifying the location in the file system where the RetailAppsObieeAdapter.properties file resides.

For example for windows environment

```
set EXTRA_JAVA_
PROPERTIES=-DRetailAppsObieeAdapter.properties=D:\Work\RetailAppsFramework\RetailA
ppsObieeAdapter.properties %EXTRA_JAVA_PROPERTIES%
```

For Unix environment

```
EXTRA_JAVA_PROPERTIES="-DRetailAppsObieeAdapter.properties
=/scratch/u00/product/Oracle/Middleware/user_projects/domains/RAFDomain/custom_
properties/RetailAppsObieeAdapter.properties ${EXTRA_JAVA_PROPERTIES}"
export EXTRA_JAVA_PROPERTIES
```

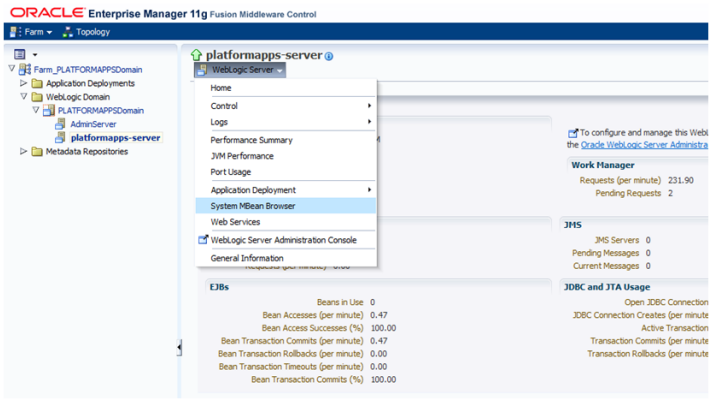
The RetailAppsObieeAdapter.properties can be used to specify the following properties for the OBIEE Report Adapter configuration.

Property	Required?	Description
bi.connection.name	Yes	This property is used to specify the name of the BISoapConnection used to display the OBIEE reports. The connection should be created using the Enterprise Manager. Please refer to the section Creating the BIConnection for more details.
bi.bipublisher.context.path	No	If the OBIEE installation includes both the OBIEE analytics and BIPublisher then the BIPublisher reports can also be displayed using the RetailAppsObieeAdapter. The bi.bipublisher.context.path is used to specify the context root of the BIPublisher installation. The default value is xmlpserver.
bi.external.integration	No	This property can be used to disable the RetailAppsObieeAdapter integration. Setting this to false will disable the RetailAppsObieeAdapter integratioin. The default value is true.
bi.report.new.browser.tab	No	This property can be used to display the OBIEE reports in a new browser tab on in a new Retail UI Shell tab. The default behavior of the RetailAppsObieeAdapter is to open the OBIEE reports in a new browser tab. Setting this property to false opens the reports in a new UI Shell tab. The default value is true.

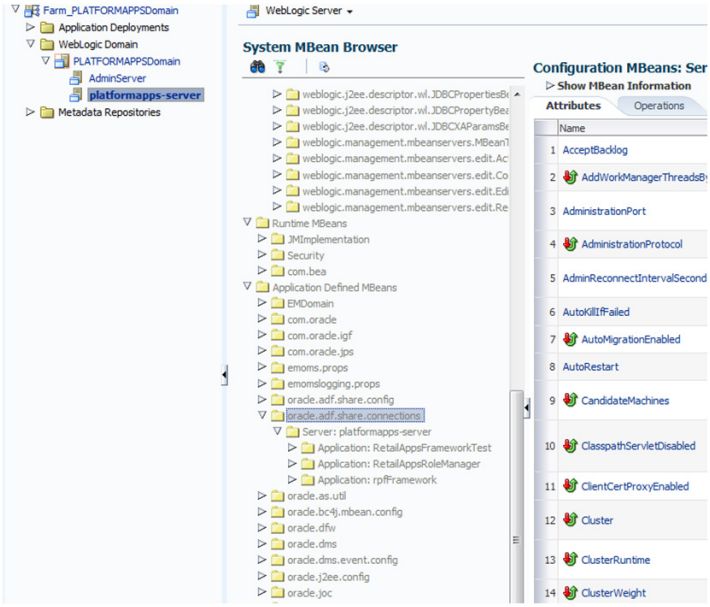
Creating the BIConnection

Please follow the following steps to create the BIConnection for the RetailAppsObieeAdapter configuration.

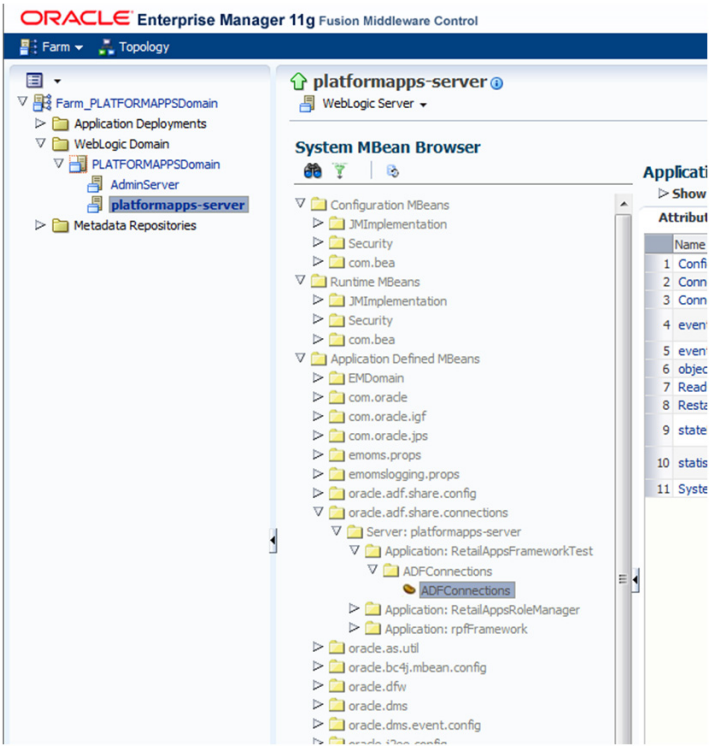
1. Login to the Enterprise Manager of the environment where the Retail Application is deployed and go to 'System MBean Browser' as shown below.



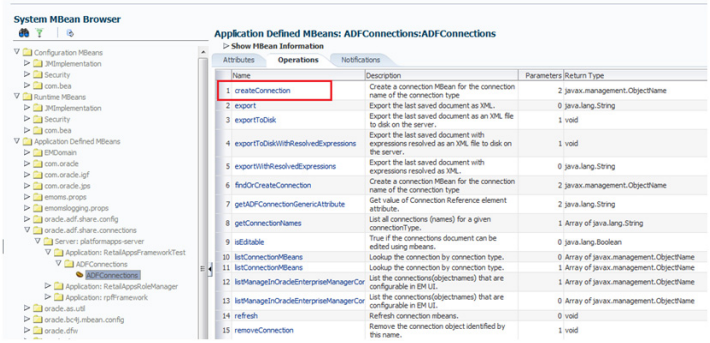
2. Go to 'Application Defined MBeans' and its sub folder 'oracle.adf.share.connections'



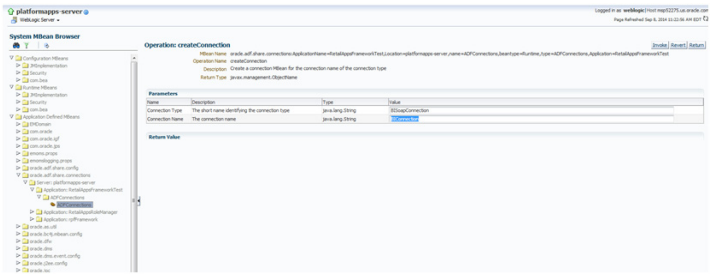
3. Expand 'Application: <AppName>' and select 'ADFConnections'. The following example uses RetailAppsFrameworkTest as the AppName.



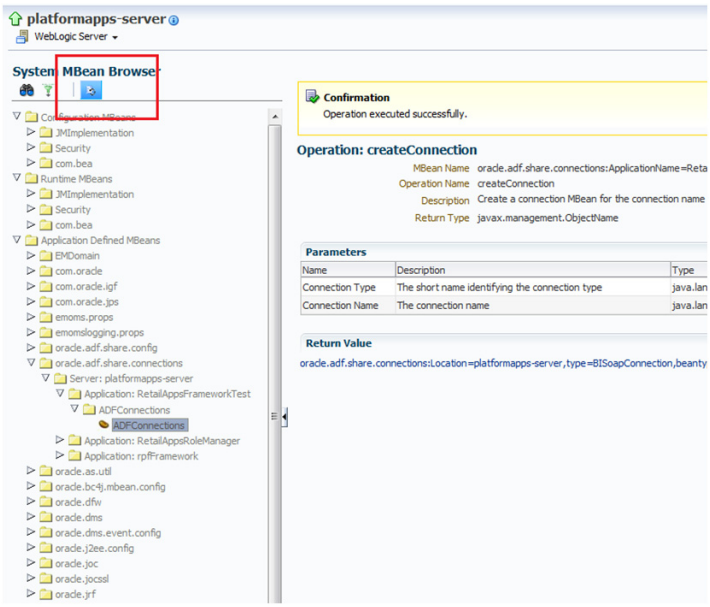
4. Go to the 'Operations' tab and click 'createConnection'.



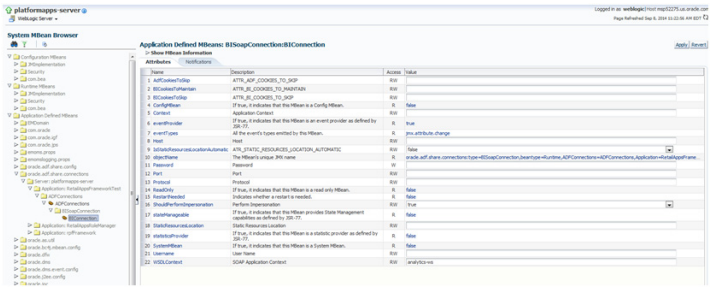
5. Enter Connection Type as BISoapConnection and Connection Name as BIConnection. Click Invoke on top right.



6. Refresh the tree using the button 'Refresh cached tree data'.



7. Once the tree is refreshed, you will notice that 'ADFConnections' becomes a folder and it has a new connection 'BIConnection'. Click on the BIConnection to open its properties in the right pane.

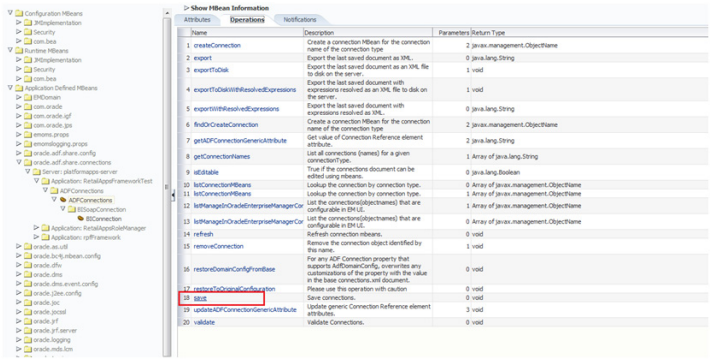


8. Enter the values shown in the table below. Once done click Apply on the top right.

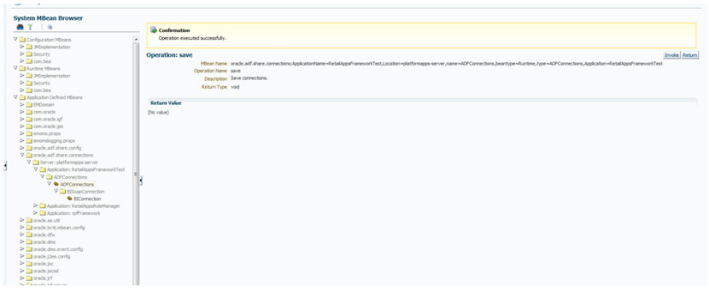
Context	The context root of the Oracle BI EE installation. Example: analytics
Host	The host of the Oracle BI EE installation.
IsStaticResourcesLocationAutomatic	The IsStaticResourcesLocationAutomatic attribute can be either true or false. When this attribute is true, the BI Presentation Server will be used to specify the location of the static resources. When this attribute is false, the attribute 'StaticResourcesLocation' should be provided. The attribute StaticResourcesLocation will be used to find the static resources when the attribute IsStaticResourcesLocationAutomatic is false.
Password	The password for the BIImpersonateUser.
Port	The port of the Oracle BI managed server. Example 9704

Protocol	http or https For production environments, use https.
ShouldPerformImpersonation	true (keep it true, default is true)
StaticResourcesLocation	The path of the location where the static resources are available. If the IsStaticResourcesLocationAutomatic attribute is true then use the following URL for the StaticResourcesLocation attribute. **http://<obiee_host>:<obiee_port>/analytics** If the IsStaticResourcesLocationAutomatic attribute is false, then use the following URL for the StaticResourcesLocation attribute. **http://<web_tier_host>:<web_tier_port>** The Oracle BI EE static resources, namely the 'res' and 'analyticsRes' directories should be available from the Oracle Web Tier's htdocs directory when the IsStaticResourcesLocationAutomatic attribute is false.
Username	The username that has been configured on the Oracle BI EE with the impersonate permissions. Example: BIImpersonateUser
WSDLContext	analytics-ws

9. Go to 'ADFConections' to save the changes. Click on 'save' operation in the Operations tab.



10. Click on 'Invoke' on the top right to save the changes.



Once the connection is created successfully, modify the RetailAppsObieeAdapter.properties to add the bi.connection.name as BIConnection.

Configuring the Reports Menu

Once we have configured the RetailAppsObieeAdapter.properties and created a BIConnection, we can add an Item of type="dynamicContent" in the HomeReportsMenuModel.xml file to display the OBIEE Reports in the Reports menu. The implementation of the DynamicContentHandler interface for OBIEE is called oracle.retail.apps.framework.report.obiee.handler.ObieeDynamicContentHandler.

```
        <Item id="obieeFolder" visible="#{true}" title="OBIEE" type="folder">
            <Items>
                <Item id="obieeDynamicContent" title="OBIEE Dynamic Menu"
type="dynamicContent"

dynamicContentHandler="oracle.retail.apps.framework.report.obiee.handler.ObieeDyna
micContentHandler"
                      visible="#{securityContext.authenticated}"/>
            </Items>
        </Item>
```

The reports menu model can be customized to add an entry for the dynamic content as shown above. Please refer to the section Adding or Modifying an Item in the Reports Menu for more details.

BIPublisher Report Adapter

The RetailAppsBiPublisherAdapter relies on a properties file called RetailAppsBiPublisherAdapter.properties for the BIPublisher configuration. Create RetailAppsBiPublisherAdapter.properties in any location on the file system. Modify the setDomainEnv.cmd for the Windows environment or setDomainEnv.sh file for the Unix environment to add the RetailAppsBiPublisherAdapter.properties as a Java system property specifying the location in the file system where the RetailAppsBiPublisherAdapter.properties file resides.

For example for windows environment:

```
set EXTRA_JAVA_
PROPERTIES=-DRetailAppsBiPublisherAdapter.properties=D:\Work\RetailAppsFramework\R
etailAppsBiPublisherAdapter.properties %EXTRA_JAVA_PROPERTIES%
```

For Unix environment:

```
EXTRA_JAVA_
PROPERTIES="-DRetailAppsBiPublisherAdapter.properties=/scratch/u00/product/Oracle/
Middleware/user_projects/domains/RAFDomain/custom_
properties/RetailAppsBiPublisherAdapter.properties ${EXTRA_JAVA_PROPERTIES}"
export EXTRA_JAVA_PROPERTIES
```

The RetailAppsBiPublisherAdapter.properties can be used to specify the following properties for the BIPublisher Report Adapter configuration.

Property	Required?	Description
bipublisher.wsdl.location	Yes	This property is used to specify the WSDL URL of the BIPublisher PublicReportService. For example: https://bipublisher_host:bipublisher_port/ xmlpserver/services/PublicReportService?wsdl
bipublisher.external.integration	No	This property can be used to disable the RetailAppsBiPublisherAdapter integration. Setting this to false will disable the RetailAppsBiPublisherAdapter integratioin. The default value is true.
bipublisher.report.new.browser.tab	No	This property can be used to display the BIPublisher reports in a new browser tab on in a new Retail UI Shell tab. The default behavior of the RetailAppsBiPublisherAdapter is to open the BIPublisher reports in a new browser tab. Setting this property to false opens the reports in a new UI Shell tab. The default value is true.

Store the BIPublisher Admin Credentials

The RetailAppsBiPublisherAdapter requires the admin credentials of the BIPublisher to be stored in the WebLogic domain credential store where the Retail Application is deployed. The admin credentials are used by the RetailAppsBiPublisherAdapter to connect to the BIPublisher.

The admin credentials can be stored the in domain credential store by using WebLogic Scripting tool or the Enterprise Manager.

WebLogic Scripting Tool

After running the wlst script, connect to the WebLogic Domain using the connect() command and run the following createCred() command. Replace the <bip_admin_user> and <bip_admin_password> with the BIPublisher admin user and password.

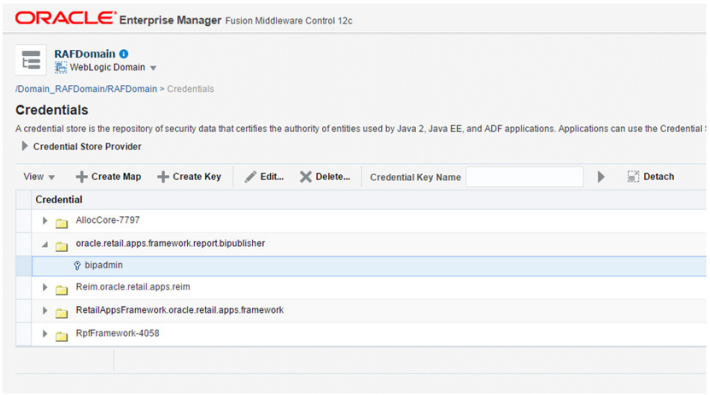
```
createCred(map="oracle.retail.apps.framework.report.bipublisher", key="bipadmin",
user="<bip_admin_user>", password="<bip_admin_password>", desc="BIPublisher admin
user credentials")
```

If the credentials already exist for the map 'oracle.retail.apps.framework.report.bipublisher' and key 'bipadmin', update them using the following command

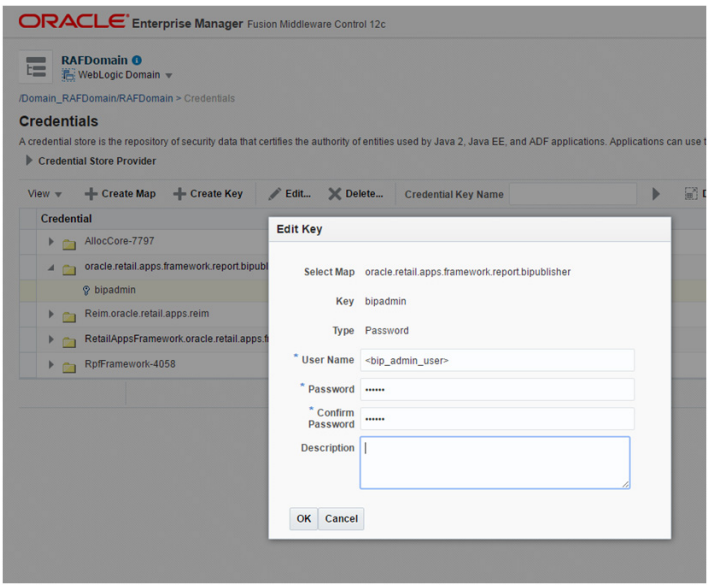
```
updateCred(map="oracle.retail.apps.framework.report.bipublisher", key="bipadmin",
user="<bip_admin_user>", password="<bip_admin_password>")
```

Enterprise Manager

1. Navigate to Security/Credentials.



2. Create a new map called 'oracle.retail.apps.framework.report.bipublisher'. Inside this map create a new key called 'bipadmin' and enter the BIPublisher admin credentials.



Configuring the Reports Menu

Once we have configured the RetailAppsBiPublisherAdapter.properties and have added the BIPublisher WSDL URL property in the properties file, we can add an Item of type="dynamicContent" in the HomeReportsMenuModel.xml file to display the BIPublilsher Reports in the Reports menu. The implementation of the DynamicContentHandler interface for the BIPublisher is called oracle.retail.apps.framework.report.bipublisher.handler.BiPublisherDynamicContentHandler.

```
                <Item id="bipublisherFolder" visible="#{true}" title="BiPublisher"
type="folder">
            <Items>
                <Item id="bipublisherDynamicContent" title="BIPublisher Dynamic
Menu"
                      type="dynamicContent"

dynamicContentHandler="oracle.retail.apps.framework.report.bipublisher.handler.BiP
ublisherDynamicContentHandler"
                      visible="#{securityContext.authenticated}"/>
            </Items>
```

```
</Item>
```

The reports menu model can be customized to add an entry for the dynamic content as shown above. Please refer to the section Adding or Modifying an Item in the Reports Menu for more details.

8

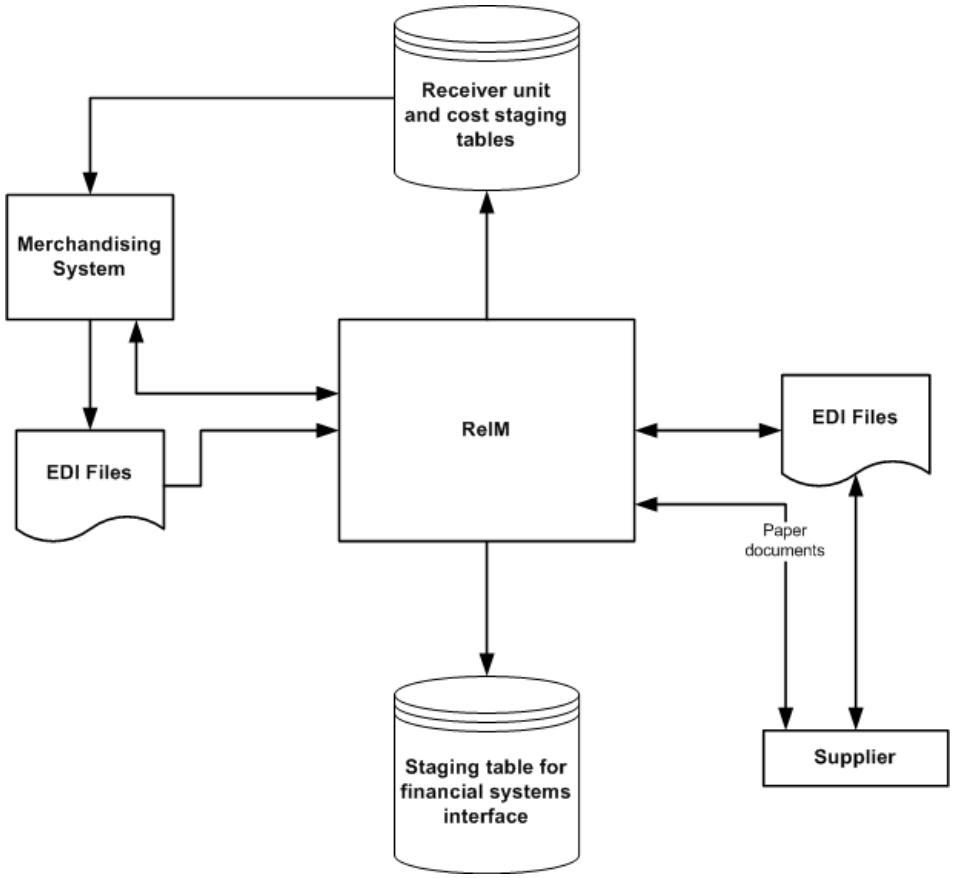
Integration

This chapter describes how ReIM integrates with other systems, related interfaces, and file layouts. It includes an integration overview, a discussion of EDI (with layouts), an explanation of how ReIM interfaces with financial systems, and a summary of LDAP user authentication.

Additional configuration may be needed to properly integrate other systems with ReIM. See Chapter 12, "System Configuration" for additional information.

Integration Overview

This section provides a diagram that shows the overall direction of the data among the applications and tables. The accompanying explanations are written from a system/staging table-to-system/staging table perspective, illustrating the movement of data.



From the Supplier (to EDI) to ReIM

ReIM receives supplier invoices and credit notes through EDI or through on-line entry processes. These document types are described later in this chapter.

From ReIM (to EDI) to the Supplier

ReIM generates debit memos, credit note requests and credit memos for various reasons. Each of these documents is recorded in ReIM tables to allow for retailer reporting. Also, a ReIM process reads these tables and creates a file of these documents to support the retailer's EDI transmissions to suppliers.

From ReIM to the Staging Table for Financial Systems Interface

For a description of the data that is sent through this interface, see "Financial System Interface" later in this chapter.

From the Merchandising System to ReIM (Directly and Through EDI)

ReIM is able to access foundation data, such as item, purchase order, supplier, and other information directly from RMS tables. ReIM provides the drivers to access these tables without further integration work.

- **Receipts**

  Receipts are records of purchased merchandise arriving at the store or warehouse. Receipt data is accessed in RMS, and certain data elements are extracted from RMS into ReIM tables to support ReIM-specific actions performed against receipts (for example, splitting receipt quantities, updating statuses, and so on).

- **Purchase Orders**

  Purchase orders (POs) are created in RMS and represent a legally binding agreement between retailer and supplier for the purchase and sale of goods. The retailer records the quantity, cost and delivery location of items from the supplier. On a single PO, RMS supports different costs for the same item going to different locations. PO costs are used to value receipt quantities.

- **Item**

  ReIM processes matches at the item transaction-level (that is, SKUs). For reference purposes, UPCs may be used, so they should be provided by the merchandising system. See Oracle Retail Merchandising System documentation for more information about the multi-level item structure.

- **Partner**

  A partner is a business that supplies and bills a retailer for non-merchandise services. Examples of partners are banks, agents, and expense suppliers. A partner cannot send merchandise invoices to retailers.

- **Valued Added Tax (TAX) Code and Rate**

  TAX is embedded in the cost of the item. ReIM provides for validation of taxes charged on the invoice against TAX codes/rates stored in RMS tables for the item.

- **Consignment**

  Consignment is an arrangement whereby the physical control of merchandise (but not the title of ownership) is transferred from one business known as the consignor (for example, the vendor) to another known as the consignee (for example, the retailer). The title to the goods remains with the consignor until the

goods are sold. When consigned goods are sold, the consignor invoices the consignee. On this invoice, the cost of each item is reduced to a certain proportion, called the consignment rate. The consignment rate, predetermined by both parties, represents the consignor's share of the sale. Once the merchandising system records a sale, a consignment invoice is created in ReIM for a percentage of the sale cost. The receipt is implied based on the consignment rate applied to the selling price; accordingly, the self-billed invoice is assumed to be in matched status.

- **Return to Vendor (RTV)**

  An RTV is a retailer-initiated purchase return of inventoried goods to an external vendor. The merchandising system uses RTV data to update inventory positions and write requisite transactions to the stock ledger. ReIM receives RTV data through the merchandising system from the store and warehouse inventory systems where it is initiated, where a charge-back document is created.

- **Deal Bill Backs**

  RMS tracks certain types of supplier deals (for example, rebates, vendor funded markdowns, and so on) for billing back to the supplier. Information to support these billings is received in ReIM through an RMS extract. ReIM creates a charge back document for these billings, which may be subject to edit/approval in ReIM or automatically processed to the financial staging table for export to the retailer's accounts payable solution, based on an RMS parameter.

- **Other Data Elements Received from RMS**
  - Non-merchandise codes
  - Currency
  - Exchange rates
  - Store/warehouse location type
  - Supplier information
  - Supplier address (invoice address, returns address, and so on)
  - Merchandise hierarchy
  - Business date
  - Terms and terms ranking

From ReIM to Receiver Unit and Cost Staging Tables to RMS

Receiver cost and unit adjustments are initiated in ReIM update receipts held in RMS tables. Receiver adjustments, resulting from the ReIM discrepancy resolution process, create cost and/quantity adjustments to receipt tables in RMS, as well as to supplier and purchase order tables for certain types of cost resolutions.

From ReIM to the Merchandising System

- **Receipt Status**

  When the entire receipt is matched (all the lines to invoices), ReIM provides and update to the invoice match status (that is, from unmatched to matched) on the shipment table in RMS.

- **Shipment (Receipts) Table Quantity Matched Update**

When ReIM matches a portion and/or all of a receipt line to an invoice line, ReIM makes a corresponding update to the quantity matched column.

Electronic Data Interchange (EDI) Tables and Files

Electronic Data Interchange (EDI) facilitates the computer-to-computer transmission of business information and transactions, such as invoices and purchase orders. EDI represents a convenient method by which a retailer and its suppliers can transfer information back and forth. The Voluntary Interindustry Commerce Standard (VICS) EDI is used by the general merchandise retail industry.

ReIM has two file-based EDI interfaces. Note that neither follows the VICS EDI standard. The ReIM EDI interfaces have been customized, and the retailer must translate them.

The interfaces represent the upload of invoices or other documents from a supplier or another application and the download of documents to suppliers. These two common types of EDI are described below:

- EDI Injector is the standard description for an EDI process that uploads documents.
- EDI invoice download is the standard description for an EDI process that downloads Debit Memo, Credit Note Request, and Credit Memo data from ReIM to suppliers.

For information about ReIM batch processes related to both of these types of EDI, see "Chapter 13, "Batch Processes". Note that although the majority of invoices are created through either EDI Injector or batch entry, users can also create invoices online and add details, or use the online dialog to add details to an invoice that was EDI uploaded.

The EDI Reject Table

The EDI Injector (reimediinjector) batch process uploads invoices and credit notes from EDI files into the Injector workspace tables (IM_INJECT_xxx), validates the data against a set of rules, and then moves valid documents into the operational tables. This process validates the information in the file against itself and against RMS/ReIM database. A limited set of data validation errors can cause the invalid transaction to remain in the Injector workspace tables (IM_INJECT_DOC_xxx) in fixable status where the data can be corrected through an online process.

The following errors can be manually corrected through the front end:

- Supplier number (or Partner ID): This value must be a valid supplier site (SUPS table) or partner (PARTNER table) in RMS (or the equivalent merchandising system).
- Order numbers: Order numbers must be approved and created for the supplier or linked suppliers in RMS (or the equivalent merchandising system) on the ORDHEAD table. Non-merchandise invoices may not have any order numbers associated, so this validation should be skipped for this type of invoice.
- Order/location combination: The system validates that all order number/location combinations in the file are valid within RMS or the equivalent merchandising system (meaning that the relationship must exist on the ORDLOC table).
- Terms code: All terms must exist within RMS or the equivalent merchandising system on the TERMS table.

- Invoice date: A document cannot be older than the VDATE minus the post-dated document days' system level parameter value or newer than the VDATE.
- Merchandise invoices cannot be associated with a partner; they must only be associated with a supplier.
- Credit notes from a partner cannot have item records attached unless the partner type is a manufacturer, distributor, or wholesaler (type S1, S2, or S3).

The EDI Reject File

A limited set of data validation errors (identified in the file layout Validation column) cause the invalid transaction to be written to the reject file (named by the retailer).

EDI Injector File Layout (Based on EDI 810)

The following describes the input and output specification for the EDI Injector File.

All Files Layouts Input and Output

Input file format:

FHEAD (1): Start of file.

THEAD (1…n): Transaction (document) level info. Each file must have at least 1 THEAD.

TDETL (0…n): Item detail records for this transaction. TDETL is required for Merchandise Invoices and optional for Non-Merchandise Invoices, Credit Note documents, and debit memo documents.

TALLW (0…n): Allowance records for this item. TALLW is optional.

TNMRC (0…n): Non-merchandise records for this transaction. Required on non-merchandise documents.

TVATS (0…n): VAT breakdown by VAT code. TVATS is optional.

TTAIL (1…n): Marks the end of a THEAD record. Each THEAD requires exactly one TTAIL.

FTAIL (1): Marks the end of the file.

TDETL and TNMRC do not need to occur in order. TALLW must follow TDETL

If records are encountered in any order other than specified above, execution of program will halt.

Example:

FHEAD

THEAD

TNMRC

FTAIL (no TTAIL encountered)

If a record descriptor is encountered other than those specified in this document, execution of program will halt.

Reject file will have an identical format. If no records are rejected, it will consist of only the FHEAD and FTAIL lines.

All character variables should be right-padded with blanks and left justified; all numerical variables should be left-padded with zeroes and right-justified. Null variables should be blank.

Single location invoices will be inserted into IM_DOC_HEAD, IM_INVOICE_DETAIL and IM_DOC_NON_MERCH.

It is assumed all values that have dependent information included in the file (for example, location has dependent information of order no, upc, upc-supp, and so on) are valid for the RMS system. The following is never anticipated to happen: only locations A, B, and C exist in RMS; EDI reads a transaction that has location D. This sort of file may not be flagged as invalid in any way.

Uploaded documents with details must have at most one associated UPC, item or VPN identifier. When system Tax processing is enabled, documents that fail to meet these criteria will be rejected to the file by the EDI Injector Batch process. When Tax is disabled, the document will be available for review and correction through the Invoice Matching user interface in the EDI Maintenance screen.

FHEAD - File Header. First record of an upload file.

Field Name	Field Type	Description	Req	Validation
Record Descriptor	Char(5)	Describes file record type	Y	Halt execution if not FHEAD.
Line id	Number(10)	Sequential file line number.	Y	Halt execution if not 0000000001.
Gentran ID	Char(5)	The type of transaction this file represents.	Y	Halt execution if not UPINV.
Current date	Char(14)	File date in YYYYMMDDHH24MISS format.	Y	Halt execution if invalid date format.

THEAD - Transaction Header. Start of a document transaction.

Field Name	Field Type	Description	Req	Validation
Record descriptor	Char(5)	Describes file record type	Y	THEAD
Line id	Number(10)	Sequential file line number	Y	Halt execution if not in sequence
Transaction number	Number(10)	Sequential transaction number. All records within this transaction will also have this transaction number.	Y	Reject entire file if: transaction number is not numeric or not in sequence first transaction number is not 0000000001

Field Name	Field Type	Description	Req	Validation
Document Type	Char(6)	Describes the type of document being uploaded. The document type will determine the types of detail information that are valid for the document upload. Stored in IM_DOC_HEAD.TYPE. Valid values are: MRCHI - Merchandise Invoice NMRCHI - Non Merchandise Invoice CRDNT - Credit Note DBMC - Debit Memo Cost DBMQ - Debit Memo Qty CRDMC - Credit Memo Cost CNRC- Credit Note Request Cost CNRQ- Credit Note Request Qty	Y	Reject transaction to file if document type is null document type is not MRCHI (merchandise invoice), NMRCHI (non-merchandise invoice), CRDNT (credit note), DBMC (Debit Memo-Cost), DBMQ (Debit Memo-Qty), CNRC (Credit Note Request-Cost), CNRQ (Credit Note Request-Qty), CRDMC (Credit Memo Cost) document type is CRDNT (credit note); vendor is not a supplier, manufacturer, distributor, or wholesaler. document type is CRDNT and TALLW records exist document type is MRCHI and item detail records DO exist for this transaction (this type of transaction must have no item detail records) document type is CRDNT,NMRCHI, DBMC, DBMQ, CRDMC, CNRC, CNRQ and any error occurs with the document
Vendor Document Number	Char(30)	Vendor's document number. Stored in IM_DOC_HEAD.EXT_DOC_ID with all characters converted to their upper case (for example, ThisDocId -> THISDOCID).	Y	Reject entire upload file if the same vendor document number occurs more than once in the file. Reject transaction to file if: ■ Vendor document number is null. ■ Vendor document number is not unique for this vendor. ■ Vendor document number is not alphanumeric and the property INVOICE_NUMBER_VALIDATION_REGULAR_EXPRESSION in reim.properties is commented out. ■ Vendor document number contains special characters that are not specified in the property INVOICE_NUMBER_VALIDATION_REGULAR_EXPRESSION in reim.properties and the property is not commented out. ■ Vendor document number has leading zero and the property INVOICE_NUMBER_VALIDATION_ALLOW_ZERO in reim.properties is commented out or set to false.

Field Name	Field Type	Description	Req	Validation
Group ID	Char(10)	The Group ID is an informational field, which can be used to identify groups of invoices that were transmitted to ReIM together.	N	Reject transaction to file if: ■ Group ID exists and is non-numeric. ■ Group ID exists and is numeric and negative.
Vendor Type	Char(6)	Type of vendor (either supplier or partner) for this document. Stored in IM_DOC_ HEAD.VENDOR_TYPE Valid values are: SUPP - Supplier BK - Bank AG - Agent FF - Freight Forwarder IM - Importer BR - Broker FA - Factory AP - Applicant CO - Consolidator CN - Consignee S1 - Merch Supp level 1 S2 - Merch Supp level 2 S3 - Merch Supp level 3	Y	Reject transaction to file if: ■ Vendor type is null or if it is not a valid vendor type (from Vendor class). ■ Document type is MRCHI (merchandise invoice) and vendor type is not Supplier (SUPP).
Vendor ID	Char(10)	Vendor for this document (if a supplier, this field must be a supplier site) . Stored in IM_DOC_ HEAD.VENDOR	Y	Reject transaction to tables if: ■ Vendor is a supplier and supplier site is not valid. ■ Vendor is a supplier and vendor ID is not completely numeric.
Vendor Document Date	Char(14)	Date document was issued by the vendor (in YYYYMMDDHH24MISS format). Stored in IM_ DOC_HEAD.DOC_DATE	Y	Reject transaction to file if: ■ Vendor document date is null. ■ Date is not a valid date format. Reject transaction to tables if Vendor Document Date is: ■ After the vdate. ■ Before (vdate - post_dated_doc_ days) (from im_system_options).

Field Name	Field Type	Description	Req	Validation
Order Number/ RTV Order Number	Number(12)	Merchandising system order number for this document. Required for merchandise invoices and optional for others. Store in IM_DOC_HEAD.ORDER_NO. This field can also contain the RTV order number if the RTV flag is 'Y'	N	Reject transaction to file if: ■ Order/RTV order number exists and is not numeric. ■ Order/RTV order number is null and vendor type is a supplier. ■ Order/RTV order number is null and deal_id is null. ■ Order/RTV order number exists and vendor type is NOT a supplier. ■ Order/RTV order number exists and location or location type are null. Reject transaction to tables if RTV flag is null or 'N' AND: ■ Order number exists but is not valid for the supplier or the supplier's linked suppliers. ■ Order number exists but is not valid for the location/location type. Reject transaction to file if RTV flag is 'Y' AND: ■ RTV order number exists but is not valid for the supplier or the supplier's linked suppliers. ■ RTV order number exists but is not valid for the location/location type.
Location	Number(10)	Merchandising system location for this document. Required for merchandise invoices and optional for others. Stored in IM_DOC_HEAD.LOCATION.	Y	Reject transaction to file if: ■ Location or location type do not exist. ■ Location exists and is not numeric. ■ Location exists and location type is not Store or Warehouse. Reject transaction to tables if Location and Location Type exist but are not valid.
Location Type	Char(1)	Merchandising system location type (either Store or Warehouse) for this document. Required for merchandise invoices and optional for others. Stored in IM_DOC_HEAD.LOC_TYPE.	N	Reject transaction to file if Location type exists and location is null.

Field Name	Field Type	Description	Req	Validation
Terms	Char(15)	Terms of this document. If terms are not provided, the vendor's default terms are associated with this record. Stored in IM_DOC_HEAD.TERMS. This value is used to get the Terms Discount Percentage to be stored on IM_DOC_HEAD.TERMS_DSCNT_PCT.	N	Reject transaction to tables if Terms exist and are not valid.
Due Date	Char(14)	Date the amount due is due to the vendor (YYYYMMDDHH24MISS format). If due date is not provided, default due date is calculated based on vendor and terms. Stored in IM_DOC_HEAD.DUE_DATE.	N	Reject transaction to file if: ■ Due date exists and is not a valid date format. ■ Due date is before the vendor document date.
Payment method	Char(6)	Method for paying this document. Stored in IM_DOC_HEAD.PAYMENT_METHOD.	N	Reject transaction to file if Payment method exists and is not valid.
Currency code	Char(3)	Currency code for all monetary amounts on this document. Stored in IM_DOC_HEAD.CURRENCY_CODE.	Y	Reject transaction to file if: ■ Currency code is null. ■ Currency code is not valid. ■ Order number exists and currency code does not match the order's currency.
Exchange rate	Number (12,4)	Exchange rate for conversion of document currency to the primary currency. Stored in IM_DOC_HEAD.EXCHANGE_RATE.	N	Reject transaction to file if Exchange rate exists and is not numeric.
Sign Indicator	Char(1)	Indicates either a positive (+) or a negative (-) total cost amount.	Y	Reject transaction to file if sign indicator is null or if it is not + or -.
Total Cost	Number(20,4)	Total document cost, including all items and costs on this document. This value is in the document currency. Stored in IM_DOC_HEAD.TOTAL_COST and IM_DOC_HEAD.RESOLUTION_ADJUSTED_TOTAL_COST.	N	Reject transaction to file if: ■ Total cost is null. ■ Total cost is not numeric. ■ Total cost does not equal the sum of extended costs for all item detail records in this transaction. ■ Total cost is not negative and vendor document type is CRDNT.
Sign Indicator	Char(1)	Indicates either a positive (+) or a negative (-) total Tax amount.	Y	Reject transaction to file if sign indicator is null or if it is not + or -.

Field Name	Field Type	Description	Req	Validation
Total VAT Amount	Number(20,4)	Total VAT amount, including all items and costs on this document. This value is in the document currency.	N	Treat as zero if null. Reject transaction to file if: ■ Total Tax amount is not null but is not numeric. ■ Total Tax amount does not equal the sum of Tax for all item detail records PLUS the sum of Tax for all non-merch items in this transaction PLUS the sum of Tax for all allowances in this transaction.
Sign Indicator	Char(1)	Indicates either a positive (+) or a negative (-) total Tax amount.	Y	Reject transaction to file if sign indicator is null or if it is not + or -.
Total Quantity	Number(12,4)	Total quantity of items on this document. This value is in EACHES (no other units of measure are supported in ReIM). Stored in IM_DOC_ HEAD.TOTAL_QTY and IM_DOC_ HEAD.RESOLUTION_ ADJUSTED_TOTAL_QTY.	Y	Reject transaction to file if: ■ Total quantity is null. ■ Total quantity is not numeric. ■ Total quantity does not equal the sum of quantities for all item detail records in this transaction. ■ Total quantity is not zero when vendor document type is 'NMRCHI'.
Sign Indicator	Char(1)	Indicates either a positive (+) or a negative (-) total Tax amount.	Y	Reject transaction to file if sign indicator is null or if it is not + or -.
Total Discount	Number(12,4)	Total discount applied to this document. This value is in the document currency. Stored in IM_ DOC_HEAD.TOTAL_ DISCOUNT	Y	Reject transaction to file if: ■ Total discount is null. ■ Total discount is not numeric.
Freight Type	Char(6)	The freight method for this document.	N	Reject transaction to file if Freight type exists and is not valid.
Paid Ind	Char(1)	Indicates if this document has been paid. Stored in IM_DOC_ HEAD.MANUALLY_ PAID_IND.	Y	Reject transaction to file if: ■ Paid ind is null. ■ Paid ind is not Y or N.
Consignment indicator	Char(1)	Y	Y	Reject transaction to file if: ■ Consignment indicator is null ■ Consignment indicator is not Y or N Do not reject transaction to table if Consignment is Y.
Deal Id	Number(10)	Deal Id from RMS if this invoice is a deal bill back invoice.	N	If Deal Id is not null, Deal Approval indicator must be 'M' or 'A'. Do not reject transaction to table if deal id is not null.
Deal Detail Id	Number(10)	Deal Detail Id from RMS and is stored in IM_DOC_ HEAD. DEAL_DETAIL_ID	N	

Field Name	Field Type	Description	Req	Validation
Deal Approval Indicator	Char(1)	Indicates if the document on IM_DOC_HEAD is to be created in Approved or Submitted status.	N	Reject to file if not blank, 'M' Submitted status or 'A' approved. Do not reject transaction to table if value is not null.
RTV indicator	Char(1)	Indicates if this invoice is a RTV invoice.	Y	Reject transaction to file if: ■ RTV indicator is null ■ RTV indicator is not Y or N Do not reject transaction to table if RTV is Y.
Custom Document Reference 1	Char(90)	This optional field is included in the upload file for client customization. No validation is performed on this field. Stored in IM_DOC_HEAD.CUSTOM_REF_1.	N	
Custom Document Reference 2	Char(90)	This optional field is included in the upload file for client customization. No validation is performed on this field. Stored in IM_DOC_HEAD.CUSTOM_REF_2.	N	
Custom Document Reference 3	Char(90)	This optional field is included in the upload file for client customization. No validation is performed on this field. Stored in IM_DOC_HEAD.CUSTOM_REF_3.	N	
Custom Document Reference 4	Char(90)	This optional field is included in the upload file for client customization. No validation is performed on this field. Stored in IM_DOC_HEAD.CUSTOM_REF_4.	N	
Cross-reference document number	Number(10)	Document that a credit note is for. Blank for all document types other than merchandise invoices. Stored in IM_DOC_HEAD.REF_DOC.	N	Reject transaction to file if Cross-reference document number exists and is not numeric
Reference Invoice Number	Char(50)	This optional field is included in the upload file for the reference of original invoice. This value is stored in IM_DOC_HEAD.REF_INV_EXT_DOC_ID	N	
Reference Credit Note Request Number	Char(50)	This optional field is included in the upload file for the reference of original credit note request. This value is stored in IM_DOC_HEAD. REF_CNR_EXT_DOC_ID	N	

TVATS - VAT breakdown by VAT code. This information is inserted in IM_DOC_TAX

Field Name	Field Type	Description	Req	Validation
Field record descriptor	Char(5)	Marks costs at Tax rate line.	Y	TVATS Reject entire transaction to file if this type of record exists and the transaction has any error. See technical design for additional validations. Reject to file if in im_system_options Tax is on, but there is no TVATS.
Line id	Char(10)	Sequential file line number.	Y	Halt execution if not in sequence.
Transaction number	Number(10)		Y	Reject entire file if: ■ Transaction number is not numeric. ■ Transaction number is not the same as the current transaction.
VAT code	Char(6)	VAT code that applies to cost.	Y	Reject to file if VAT code is not valid.
VAT rate	Number(20,10)	VAT Rate corresponding to the Tax code.	Y	Reject to file if VAT rate is not numeric.
Sign indicator	Char(1)	Indicates either a positive (+) or a negative (-) Original Document Quantity amount.	Y	Reject transaction to file if sign indicator is null or if it is not + or -.
Cost at this VAT code	Number(20,4)	Total amount that must be taxed at the above VAT code.	Y	Reject to file if not numeric.

TDETL - Item Detail Record. This information is inserted into the IM_INVOICE_DETAIL table for Merchandise Invoice and IM_DOC_DETAIL_REASON_CODES for Credit Notes.

Field Name	Field Type	Description	Req	Validation
Record descriptor	Char(5)	Describes file record type.	Y	TDETL
Line Id	Number(10)	Sequential file line number.	Y	Halt execution if not in sequence.
Transaction number	Number(10)	Transaction number for this item detail record.	Y	Reject to file if Tax rate is not numeric ■ Transaction number is not numeric. ■ Transaction number is not the same as the current transaction.

Field Name	Field Type	Description	Req	Validation
UPC	Char(25)	UPC for this detail record. Valid item number is retrieved for the UPC. Stored in IM_INVOICE_DETAIL.ITEM or IM_DOC_DETAIL_REASON_CODES.ITEM.	Y Exclusive with item	Reject transaction to file if: ■ UPC is null and Item is null. ■ Both UPC and Item are not null. Reject transaction to tables if: ■ Valid item is not found for UPC and UPC supp. ■ Valid item is not associated with the supplier. ■ The item found is identical to another detail item for this transaction (no duplicate items).
UPC Supplement	Number(5)	Supplement for the UPC. Note: UPC Supp is only valid for 9.0 implementation. For 10.1 implementation, this field will ALWAYS be blank.	N	Reject transaction to file if: ■ UPC supplement exists and UPC doesn't exist. ■ UPC supplement exists and is not numeric.
Item	Char(25)	Item for this detail record. Item number is verified and stored in IM_INVOICE_DETAIL.ITEM or IM_DOC_DETAIL_REASON_CODES.ITEM.	Y Exclusive with UPC	Reject transaction to file if: ■ UPC is null and Item is null. ■ Both UPC and Item are not null. ■ Valid item is not associated with the supplier. ■ The item found is identical to another detail item for this transaction (no duplicate items).
VPN	Char(30)	Vendor Product Number provided by the supplier. It is used to identify an item when an item number has not been provided. VPN is displayed on the Invoice Maintenance screen and may be used during the on-line matching process.	Y (exclusive with item and UPC)	Reject transaction to file if: ■ VPN is null and UPC is null and Item is null. ■ At least two of the following are not null: UPC, VPN and ITEM. Reject transaction to tables if: ■ Valid item is not found for VPN for the supplier. ■ The item found is identical to another detail item for this transaction (no duplicate items). ■ There are multiple items for the supplier with the VPN provided and: no items on the PO for the document OR multiple items on the PO for the document.
Sign Indicator	Char(1)	Indicates either a positive (+) or a negative (-) Original Document Quantity amount.	Y	Reject transaction to file if sign indicator is null or if it is not + or -.

Field Name	Field Type	Description	Req	Validation
OriginalDocument Quantity	Number(12,4)	Quantity, in EACHES, of the item on this detail record. Stored in IM_INVOICE_DETAIL.INVOICE_QTY and IM_INVOICE_DETAIL.RESOLUTION_ADJUSTED_QTY.	Y	Reject transaction to file if: ■ Original document quantity is null. ■ Original document quantity is not numeric.
Sign Indicator	Char(1)	Indicates either a positive (+) or a negative (-) Original Unit Cost amount.	Y	Reject transaction to file if sign indicator is null or if it is not + or -.
Original Unit Cost	Number(20,4)	Unit cost, in document currency, of the item on this detail record. Stored in IM_INVOICE_DETAIL.UNIT_COST and IM_INVOICE_DETAIL.RESOLUTION_ADJUSTED_UNIT_COST.	Y	Reject transaction to file if: ■ Original unit cost is null. ■ Original unit cost is not numeric.
Original VAT Code	Char(6)	Tax code for item.	Y	Reject to file if VAT code is invalid.
Original VAT rate	Number(20,10)	Tax Rate for the Tax code/item.	Y	Reject to file if VAT rate is not numeric.
Sign Indicator	Char(1)	Indicates either a positive (+) or a negative (-) Original Document Quantity amount.	Y	Reject transaction to file if: ■ Sign indicator is null. ■ Sign indicator is not + or -.
Total Allowance	Number(20,4)	Sum of allowance details for this item detail record. If no allowances exist for this item detail record, value is 0.	Y	Reject transaction to file if: ■ Total allowance is null. ■ Total allowance is not numeric. ■ Total allowance does not equal the sum of allowance amounts for all allowance records in this item detail record. ■ Total allowance is not 0 and vendor document type is CRDNT.

TALLW - Allowance Record. This information is inserted into IM_INVOICE_DETAIL_ALLOWANCE table.

Field Name	Field Type	Description	Req	Validation
Record descriptor	Char(5)	Describes file record type.	Y	TALLW
Line id	Number(10)	Sequential file line number.	Y	Halt execution if not in sequence.

Field Name	Field Type	Description	Req	Validation
Transaction Number	Number(10)	Transaction number for this item allowance record.	Y	Reject entire file if: ■ Transaction number is not numeric. ■ Transaction number is not the same as the current transaction.
Allowance Code	Char(6)	Allowance code for this allowance record. Stored in IM_INVOICE_ DETAIL_ ALLOWANCE.ALLOWANCE_CODE.	Y	Reject transaction to file if: ■ Allowance code is null. ■ Allowance code is not valid.
Sign Indicator	Char(1)	Indicates either a positive (+) or a negative (-) allowance amount.	Y	Reject transaction to file if sign indicator is null or if it is not + or -.
Allowance Amount	Number (20,4)	Amount of allowance in document currency. Stored in IM_INVOICE_ DETAIL_ ALLOWANCE.ALLOWANCE_AMOUNT.	Y	Reject transaction to file if allowance amount is null or not numeric.
Allowance VAT Code	Char(6)	VAT Code for Allowance.	Y	Reject to file if VAT code is not valid.
Allowance VAT rate at this VAT code	Number (20,10)	VAT Rate corresponding to theVAT code.	Y	Reject to file if not numeric.

TNMRC - Non-Merchandise Record. Records of this type will contain non-merchandise costs. These costs are inserted into the IM_DOC_NON_MERCH table. Non-merchandise costs records are only required when the document type is non-merchandise. Non-merchandise cost records are also associated with merchandise type documents if the vendor associated with the document allows non-merch costs on merchandise invoices (IM_SUPPLIER_OPTIONS. MIX_ MERCH_NON_MERCH_IND).

Field Name	Field Type	Description	Req	Validation
Record descriptor	Char(5)	Describes file record type.	Y	TNMRC
Line id	Number(10)	Sequential file line number.	Y	Halt execution if not in sequence.
Transaction number	Number(10)	Transaction number for this non-merchandise record.	Y	Reject entire file if: ■ Transaction number is not numeric. ■ Transaction number is not the same as the current transaction.
Non Merchandise Code	Char(6)	Non-Merchandise code that describes this cost. Stored in IM_DOC_ NON_MERCH.NON_ MERCH_CODE.	Y	Reject transaction to file if: ■ Non-merchandise code is null. ■ Non-merchandise code is not valid.
Sign Indicator	Char(1)	Indicates either a positive (+) or a negative (-) Non Merchandise Amt.	Y	Reject transaction to file if sign indicator is null or if it is not + or -.

Field Name	Field Type	Description	Req	Validation
Non Merchandise Amt	Number(20,4)	Cost in the document currency. Stored in IM_DOC_NON_MERCH.NON_MERCH_AMT.	Y	Reject transaction to file if: ■ Non-merchandise amount is null. ■ Non-merchandise amount is not numeric. ■ Non-merchandise amount does not have a negative value and this is part of a credit note document (THEAD.Vendor Document Type = CRDNT).
Non Merch VAT Code	Char(6)	VAT Code for Non-Merchandise.	Y	Reject to file if VAT code is not valid.
Non Merch VAT code at this VAT code	Number(20,10)	VAT Rate corresponding to the VAT code.	Y	Reject to file if not numeric.
Service Performed Indicator	Char(1)	Indicates if a service has actually been performed. Stored in IM_DOC_NON_MERCH.SERVICE_PERF_IND.	Y	Reject transaction to file if: ■ Service performed indicator is null. ■ Service performed indicator is not Y or N.
Store	Number(10)	Store at which the service was performed. Stored in IM_DOC_NON_MERCH.STORE.	N	Reject transaction to file if: ■ Store exists and is not numeric. ■ Service performed indicator is Y and store is not valid.

TTAIL - Transaction Tail. Marks the end of a transaction.

Field Name	Field Type	Description	Req	Validation
Record descriptor	Char(5)	Describes file record type.	Y	TTAIL
Line id	Number(10)	Sequential file line number.	Y	Halt execution if not in sequence.
Transaction number	Number(10)	Transaction number for the transaction that this record is closing.	Y	Reject entire file if: ■ Transaction number is not numeric. ■ Transaction number is not the same as the current transaction.
Transaction lines	Number(6)	Total number of detail lines within this transaction.	Y	Reject transaction to file if transaction lines is not numeric, if it does not match the count of lines within the transaction, or if it is zero (transaction must have details).

FTAIL - File TAIL. Marks the end of the upload file.

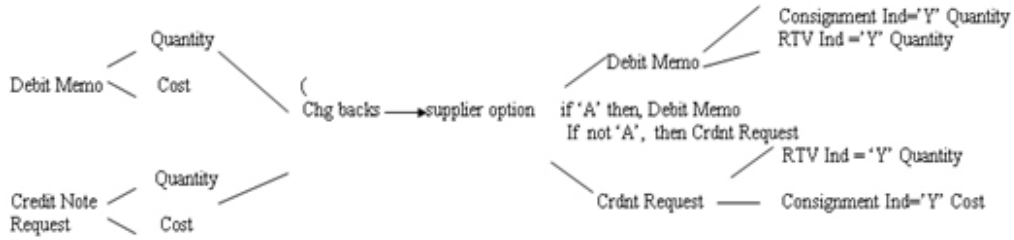
Field Name	Field Type	Description	Req	Validation
Record descriptor	Char(5)	Describes file record type.	Y	FTAIL

Field Name	Field Type	Description	Req	Validation
Line id	Number(10)	Sequential file line number.	Y	Halt execution if not in sequence.
Number of lines	Number(10)	Total number of lines within this file not counting FHEAD and FTAIL.	Y	Halt execution if number of lines is not numeric, if it does not match the count of lines within the file (excluding FHEAD and FTAIL), or if it is 2 (FHEAD and FTAIL only, file has no transactions).

Notes

Consider the following.

- The EDI Injector Batch process has the ability to recognize only a new document type. In FHEAD of the EDI flat file, the Document Type does not include CRDMC (credit memo cost). When the document type in the flat file is Debit Memo Cost, Debit Memo Qty, Credit Note Request Cost, or Credit Note Request Qty, and if the amount (Total Cost) for a Deal Charge Back Document that is sent over from RMS is negative a Credit Memo Cost is created.
- For charge back documents, use the following flow chart to determine what document type to be populated in the database.



- If the document type is merchandise invoice and the consignment indicator is Y, the status is matched; if the consignment indicator is not Y, the status isready for match; if the document type is not merchandise invoice, the status is approved.
- If the consignment indicator is Y, then set the terms to Due Immediately terms (term ID = 48), and set the terms discount percentage to 0.
- That Tax codes and rates in the detail of documents are those known for the item and location when the document is not an import Document. Given a combination of TDETL.item and location, we could find a Tax. The Tax code and Tax rate in the Tax should be the same as the original Tax code and original Tax rate in the TDETL.
- The merchandises header Tax and detail Tax are consistent (for example, Tax basis by Tax rate and Tax amount by Tax rate). Total header Merchandise Tax information is calculated from total document Tax information and Tax information for non merchandise costs. For example, for each Tax Code in TDETL and TNMRC: Thead.Total VAT Amount at this Tax code = total Tax from TDETL at this Tax code + total Tax from TNMRC at this Tax code. Total Tax from TDETL at this Tax code = sum(original document quantity * original unit cost * original Tax rate). Total Tax from TNMRC at this Tax code =sum(Non Merch Tax rate * Non Merch Amt).Thead.Total VAT Amount at this VAT code = sum(TVATS.Vat rate * TVATS.cost at this VAT code).

- For an EDI upload document, if the Tax Region of the header is different from the Tax region of the supplier, it is an import document. Import document will not contain Tax information. (LocVatRegion != SupplierVatRegion, then it is an import document). If a document is not an import document, plus the system_option.vat is on; if the TVATS is null, reject to file.
- To decide whether a Tax code is valid in the TDETL, first find the Tax code given the information of item and location. If they are equal, then the Tax code is valid; if they are not equal, check if the Tax code exists in the effective Tax codes; if the Tax code exists, then it is valid but is populated to the audit table.
- If RTV indicator or consignment indicator is Yes and deal ID is not null, it must reject to file.
- If Item field is populated and there is an error it should always reject to file. In order to reject to the tables, we must have the UPC field populated and not the Item field.

EDI Invoice Download File Layout (Based on EDI 812)

Output file format:

FHEAD (1): Start of file.

THEAD (1…n): Transaction (document) level info. Each file must have at least 1 THEAD.

TDETL (0…n): Item detail records for this transaction.

TNMRC (0…n): Non-merchandise records for this transaction.

Required on non-merchandise documents, optional otherwise.

TVATS (0…n): Doc Tax detail records for this transaction, optional.

TTAIL (1…n): Marks the end of a THEAD record. Each THEAD requires exactly one TTAIL.

FTAIL (1): Marks the end of the file.

If records are encountered in any order other than specified above, execution of program will halt.

Example:

FHEAD

THEAD

TNMRC

TVATS

FTAIL (no TTAIL encountered)

If a record descriptor is encountered other than those specified in this document, execution of program will halt.

All character variables should be right-padded with blanks and left justified; all numerical variables should be left-padded with zeroes and right-justified. Null variables should be blank.

Note: The file is not threaded, but rather ordered by vendor id (THEAD). It is assumed that this file is broken out by vendor id during the translation process.

FHEAD - File Header. First record of an upload file.

Field Name	Field Type	Description	Req	Validation
Record descriptor	Char(5)	FHEAD	Y	
Line id	Number(10)	Generated Sequential file line number.	Y	
Gentran ID	Char(5)	DNINV	Y	
Current date	Char(14)	File date in YYYYMMDDHH24MISS format.	Y	

THEAD - Transaction Header. Start of a document transaction.

Field Name	Field Type	Description	Req	Validation
Record descriptor	Char(5)	THEAD	Y	
Line id	Number(10)	Generated Sequential file line number.	Y	
Transaction number	Number(10)	Sequential transaction number. All records within this transaction will also have this transaction number.	Y	
Document Type	Char(6)	Describes the type of document being downloaded. The document type will determine the types of detail information that are valid for the document downloaded. Retrieved from IM_DOC_HEAD.TYPE where type is debit memo, credit note request or credit memo and in Approved or Posted Status.	Y	
Vendor Document Number	Char(30)	Vendor's document number. Retrieved from IM_DOC_HEAD.EXT_DOC_ID.	Y	
Invoice Number	Char(6)	Corresponding invoice resolved by the document. Retrieved from IM_DOC_HEAD.REF_DOC.	Y	
Vendor ID	Number(10)	Vendor for this document. Retrieved from IM_DOC_HEAD.VENDOR	Y	
Document Date	Char(14)	Date the document was entered into the system in YYYYMMDDHH24MISS format. Retrieved from IM_DOC_HEAD.DOC_DATE	Y	
Order	Number(10)	Order number for this document, if any. Retrieved from IM_DOC_HEAD.ORDER_NO	N	
Location	Number(10)	Location for this document, if any. Retrieved from IM_DOC_HEAD.LOCATION.	N	

Field Name	Field Type	Description	Req	Validation
Location Type	Char(1)	Location type for this document, if any. Retrieved from IM_DOC_HEAD.LOC_TYPE.	N	
Terms	Char(15)	Terms of this document. Retrieved from IM_DOC_HEAD.TERMS.	N	
Due Date	Char(14)	Date the amount due is due from the vendor (YYYYMMDDHH24MISS format). Retrieved from IM_DOC_HEAD.DUE_DATE.	N	
Currency Code	Char(3)	Currency code for this document. Retrieved from IM_DOC_HEAD.CURRENCY_CODE.	N	
Exchange Rate	Number(12,4)	Exchange rate for conversion of document currency to the primary currency. Retrieved from IM_DOC_HEAD.EXCHANGE_RATE.	N	
Sign indicator	Char(1)	Indicates either a positive (+) or a negative (-) total cost.	Y	
Total Cost	Number(20,4)	Total document cost, including all items and costs on this document. This value is in the document currency. Retrieved from IM_DOC_HEAD.TOTAL_COST.	Y	
Sign indicator	Char(1)	Indicates either a positive (+) or a negative (-) total VAT amount	Y	
Total VAT Amount	Number(20,4)	Total VAT amount, including all items and costs on this document. This value is in the document currency.	N	
Sign indicator	Char(1)	Indicates a positive (+) or negative (-) quantity	Y	
Total Quantity	Number(12,4)	Total quantity of items on this document. This value is in EACHES (no other units of measure are supported in ReIM). Retrieved from IM_DOC_HEAD.TOTAL_QTY.	Y	

TDETL - Item Detail Record. This information is inserted into the IM_DOC_DETAIL_REASON_CODES table.

Field Name	Field Type	Description	Req	Validation
Record descriptor	Char(5)	TDETL	Y	
Line id	Number(10)	Generated Sequential file line number.	Y	
Transaction number	Number(10)	Generated Transaction number for this item detail record	Y	
Item	Char(25)	Internal SKU/Item for this document. This is always sent. Retrieved from IM_DOC_DETAIL.ITEM.	Y	

Field Name	Field Type	Description	Req	Validation
UPC	Char(25)	UPC for this detail record. Retrieved from UPC_EAN.UPC (RMS 9.0) or ITEM_MASTER.ITEM (RMS 10.1). This field is sent if available. Note: UPC is used for RMS 9.0 and Ref-Item is used for RMS 10.1. Ref-Item consists of UPC and UPC-Supp appended together with a separating hyphen(-).	N	
UPC Supplement	Number(5)	Supplement for the UPC. Retrieved from UPC_EAN.UPC_SUPPLEMENT. This field is sent if available. Note: UPC Supp is only valid for 9.0 implementation. For 10.1 implementation, this field will always be blank.	N	
VPN	Char(30)	Vendor Product Number. This field is sent if available. Retrieved from ITEM_SUPPLIER.VPN.	N	
Comments	Char(200)	Comments associated with Reason Code. Retrieved from IM_DOC_DETAIL_COMMENTS.TEXT	Y	
Reason Code	Char(6)	Reason Code for this document. Retrieved from IM_DOC_DETAIL_REASON_CODES.REASON_CODE_ID	Y	
Reason Code description	Char(50)	Description associated with Reason Code. Retrieved from IM_REASON_CODES.REASON_CODE_DESC		
Sign indicator	Char(1)	Indicates a positive (+) discrepant qty.	Y	
Discrepant Quantity	Number(12,4)	Quantity, in EACHES, of the item that is discrepant for this detail record. Retrieved from IM_DOC_DETAIL_REASON_CODES.ADJUSTED_QTY.	Y	
Sign indicator	Char(1)	Indicates either a positive (+) or a negative (-) discrepant cost.	Y	
Discrepant cost	Number(20,4)	Unit cost, in document currency, of the item that is discrepant for this detail record. Retrieved from IM_DOC_DETAIL_REASON_CODES.ADJUSTED_UNIT_COST.	Y	
Original VAT code	Char(6)	VAT code for item.		
Original VAT rate	Number(20,10)	VATRate for the VAT code/item.		

TNMRC - Non-Merchandise Record. Records of this type will contain non-merchandise costs. These costs are retrieved from the IM_DOC_NON_MERCH table. Non-merchandise cost records are only required when the document type is non-merchandise. Non-merchandise cost records are also associated with merchandise type documents if the vendor associated with the document allows non-merch costs on merchandise invoices (IM_SUPPLIER_OPTIONS. MIX_MERCH_NON_MERCH_IND).

Field Name	Field Type	Description	Req	Validation
Record descriptor	Char(5)	TNMRC	Y	
Line id	Number(10)	Generated Sequential file line number.	Y	
Transaction number	Number(10)	Generated Transaction number for this non-merchandise record.	Y	
Non Merchandise Code	Char(6)	Non-Merchandise code that describes this cost. Retrieved from IM_DOC_ NON_MERCH.NON_MERCH_CODE.	Y	
Sign indicator	Char(1)	Indicates either a positive (+) or a negative (-) non merchandise amount.	Y	
Non Merchandise Amt	Number(20,4)	Cost in the document currency. Retrieved from IM_DOC_NON_MERCH.NON_ MERCH_AMT.	Y	
Non Merch VAT code	Char(6)	VAT Code for Non_merchandise.	Y	
Non Merch VAT code at this VAT code	Number(20,10)	VATRate corresponding to the VAT code.		

TVATS - VAT Detail record.

Field Name	Field Type	Description	Req	Validation
Record descriptor	Char(5)	TVATS	Y	
Line id	Number(10)	Sequential line number.	Y	
Transaction number	Number(10)		Y	
VAT code	Char(6)	VAT code that applies to cost.	Y	
VAT rate	Number(20,10)	VAT Rate corresponding to the VAT code.	Y	
Sign indicator	Char(1)	Indicates either a positive (+) or a negative (-) Original Document Quantity amount.	Y	
VAT Basis	Number(20,4)	Total amount that must be taxed at the above VAT code.	Y	

TTAIL - Transaction Tail. Marks the end of a transaction.

Field Name	Field Type	Description	Req	Validation
Record descriptor	Char(5)	TTAIL	Y	
Line id	Number(10)	Generated Sequential file line number.	Y	
Line number	Number(10)	Generated Transaction number for the transaction that this record is closing.	Y	
Transaction lines	Number(6)	Total number of detail lines within this transaction.	Y	

FTAIL - File TAIL. Marks the end of the upload file.

Field Name	Field Type	Description	Req	Validation
Record descriptor	Char(5)	FTAIL	Y	
Line id	Number(10)	Generated Sequential file line number.	Y	
Number of lines	Number(10)	Total number of lines within this file, not including FHEAD and FTAIL.	Y	

Document Induction via UI

The application allows injecting spreadsheets in ODS format with document data to be processed similarly to EDI injector flow. ODS files are spreadsheets in ODF data format. This format is a binary format and is nothing but archive (zip) of actual data in XML format.

Out of the box the application will have a set of templates defined within the system. The templates will be grouped by a category. At the same time retailers will be able to define custom templates.

Template categories will be document type based. There will be one or more Merchandising Invoice templates, one or more Credit Note templates, etc.

The purpose of templates is to define the set of worksheets that are suitable for document entry process. Each worksheet will have a set of fields. While some worksheets are mandatory for a particular document type, some may be optional and as such can be removed by customization process to better suite retailer business process. Same logic applies to worksheet fields. While some of the fields are mandatory, some are optional and can be removed by customization.

Customization will be done via backend. Retailers will be able to delete base installed template. It will be up to retailers to guarantee that all the required templates are present and correct, as well as that they will not conflict with existing templates.

Template definition will be done using existing data model.

The template type will match existing document types that are allowed to be manually entered.

- NMRCHI
- MRCHI
- DEBMEQ
- DEBMEC
- CRDNRQ
- CRDNRC
- CRDMEQ
- CRDMEC

The main difference for default template definitions will be presence or lack thereof of tax related information. Another difference will be the presence or lack thereof of detail records.

Templates are defined in the following tables

- S9T_TEMPLATE - template table

- S9T_TMPL_WKSHT_DEF - template worksheets table
- S9T_TMPL_COLS_DEF- template worksheet columns table.

The template data structure is shared by multiple applications. ReIM templates are defined under DOCS9T category.

The order of the columns within the worksheet is NOT customizable and is predefined within the application. The order of the worksheets, on the other hand can be changed.

Processing of the spreadsheets is done using the same set of rules as EDI file processing. Errors identified during processing will be included in an error worksheet within the rejection file that can be retrieved via UI.

Financial System Interface

ReIM exports data to financial staging tables. This section describes these tables.

Foundation Financial Data Overview

The following types of financial information are imported in ReIM:

- Terms ranking data
- Variable department/class account segments
- Variable company/location account segments

Terms ranking information is used in the best terms calculation to choose the best term for each document. This best terms information is posted to the financial system.

Variable department/class and company/location segments are used to determine the account segments to which a document is posted.

The retailer is responsible for populating variable department/class and company/location segments. No API is provided.

Partners can only be set up as suppliers in the Financial System. After the partner is setup as a supplier, the supplier integration step is run and the partner is created as a supplier in Merchandising. Then the partner must be manually created in Merchandising using the Merchandising Supplier ID which was generated as part of the supplier integration step. Partner functionality within Merchandising and Invoice Matching can then proceed normally. The Merchandising supplier generated as part of this process is not used.

Location Account Segments

ReIM uses location account segments in general ledger (GL) account mappings. The location account segments are accessed and maintained through the ReIM user interface. The segments are dynamically assigned during posting, based on the location on the invoice. The data is stored in the location account segments table (IM_DYNAMIC_SEGMENT_LOC).

Department/Class Account Segments

ReIM uses department/class account segments in GL account mappings. The department/class account segments are accessed and maintained through the ReIM user interface. The segments are dynamically assigned during posting, based on the item on the invoice. The data is stored in the department/class account segment table (IM_DYNAMIC_SEGMENT_DEPT_CLASS).

Financial Transactions

ReIM writes two types of transactions to the financial staging tables. Documents which are expected to be sent to the A/P system are sent to the IM_AP_STAGE_HEAD and IM_AP_STAGE_DETAIL tables. Transactions expected to go to the G/L system go to the IM_FINACIALS_STAGE table. The RFI integration will send the data in these tables to EBS or Peoplesoft. Retailers using other financial systems can still use these tables to integrate to their specific A/P and G/L systems.

Complex and Fixed Deal-Related Posting

For complex and fixed deals, batch processes copy most of the data from the RMS staging tables into ReIM detail tables (IM_COMPLEX_DEAL_DETAIL, IM_FIXED_DEAL_DETAIL). Some of the data on these tables is later referenced during the posting process for the created documents, including:

- Location
- Item

Financial Posting

To understand the process that posts data from ReIM to the financials staging table (IM_FINANCIAL_STAGE), see "Financial Posting Batch Design".

Tracking Receipt Posts

Receipt tracking functionality allows the retailer to track what receipts have posted. This processing helps the retailer check the integrity of its financial data.

Note that Oracle Retail does not provide packaged reporting in conjunction with this processing. Rather, the retailer builds its own processes and creates its own reporting mechanisms against the data resulting from the receipt tracking functionality.

Tables Related to Tracking Receipt Posts

In-Process Tables

The tables illustrated below are for the retailer's understanding, but the data on these tables should not be used by the retailer as it builds its processes and reports.

Each area of the system that matches receipts to invoices updates the IM_RECEIPT_ITEM_POSTING table. This table tracks how much of an individual receipt item has been matched and posted.

IM_RECEIPT_ITEM_POSTING

Column Type	Type	Nullable
SEQ_NO	NUMBER(10)	N
RECEIPT_ID	NUMBER(10)	N
ITEM_ID	VARCHAR(25)	N
QTY_MATCHED	NUMBER(12,4)	Y
QTY_POSTED	NUMBER(12,4)	Y

IM_RCPT_ITEM_POSTING_INVOICE

Column Type	Type	Nullable
SEQ_NO (from IM_RECEIPT_ ITEM_POSTING)	NUMBER(10)	N
DOC_ID	NUMBER(10)	N
STATUS	VARCHAR2(1)	Y

Staging Tables to be used for Reporting Once posting is completed, the following staging tables contain all currently posted entries. Thus, to build processes and reporting that tracks receipt posts, the retailer should use only the data from these staging tables.

IM_RECEIPT_ITEM_POSTING_STAGE

Column Type	Type	Nullable
SEQ_NO	NUMBER(10)	N
RECEIPT_ID	NUMBER(10)	N
ITEM_ID	VARCHAR(25)	N
QTY_POSTED	NUMBER(12,4)	N
CREATE_DATE	DATE	N

IM_RCPT_ITEM_POSTING_INV_STAGE

Column Type	Type	Nullable
SEQ_NO	NUMBER(10)	N
DOC_NO	NUMBER(10)	N

Multiple Lines for an Individual Receipt Item

For a given line item on a receipt, a line item can be split between multiple invoices. For example, one invoice could match half of a line item; another invoice could match the other half of the line item. Two separate lines would thus appear. The retailer should note that these values (and those in equivalent business scenarios) need adding together to indicate how much of a given receipt item is posted.

LDAP and Other User Interfaces

ReIM supports only LDAP user authentication. To indicate the authentication method, select LDAP for the property called authentication_source in the reim.properties file.

LDAP

Light Directory Access Protocol (LDAP) is the means of user authentication supported by ReIM. It defines a network protocol for accessing information in a directory.

Only LDAP is used within ReIM for user authentication. Because ReIM has specific requirements for ReIM user roles and permissions that are easily configurable by the retailer, they are defined in the application itself. ReIM reads standard user information from an LDAP server.

There are two types of roles; business roles and enterprise roles. Business roles are defined within the ReIM application and define what functional privileges the user

has. Enterprise roles are defined in LDAP (it is the same as user group). Any ReIM user defined in LDAP should be part of an LDAP user group defined by roles_base_dn in ldap.properties

If the retailer already stores user information using LDAP, the only interfacing configuration required is through the LDAP-specific properties file. The entries in this file point ReIM to the appropriate machine and port to find the LDAP server. Other properties also may be modified to reflect the names of attributes that the retailer uses in its LDAP schema.

Setting up the LDAP interface entails completing tasks within LDAP andReIM as follows.

Setup Steps within LDAP

In LDAP, complete the following steps.

1. Define the following attributes (or find the existing attributes that provide the same information) within your LDAP (actual name is not important).
   - User ID (standard uid).
   - Password.
   - First name (standard cn).
   - Last name (standard sn).
   - Preferred user language (identifies user locale).
   - Preferred user country (identifies user locale).
   - Preferred email (standard mail).

   Note: All attributes listed above should be single value and mandatory. For attributes with multiple values, the first value is used within ReIM.

2. Create an attribute class that encompasses all the attributes above.
3. Select a container within your LDAP to hold users, which may be created directly in the container or in enclosed containers.
4. Either create new users based on the attribute class just created above, or add the attribute class to the existing users. Define missing values as needed.

Setup Steps within ReIM

In ReIM, complete the following steps.

1. In reim.properties, change authentication_source to LDAP.
2. In ldap.properties, define the values for the parameters shown in the following table.

Parameter	Description
connection_url	Machine and port for your LDAP server
ldap_alias	Alias for the wallet entry containing the credentials for the user defined within LDAP that has ADMIN privileges

Parameter	Description
base_dn	Name of the container for ReIM users
roles_base_dn	Name of the container for ReIM role
login_id_attribute_name	Name for user ID attribute used within the attribute class
user_container	Name of the container for ReIM users
user_first_name_attribute_name-	Name for the first name attribute used within the attribute class
user_last_name_attribute_name	Name for the last name attribute used within the attribute class
user_email_attribute_name-	Name for the email attribute used within the attribute class
user_password_attribute_name-	Name for the password attribute used within the attribute class
user_language_attribute_name	Name for the preferred language attribute used within the attribute class
user_country_attribute_name-	Name for the preferred country attribute used within the attribute class
user_main_key	User attribute which stores the username to be used in ReIM
role_member	Group attribute which stores the distinguished name of users in the group
role_application	Group attribute which stores the group name.
WildCard	Wild character used to identify any search criteria filter
referral_handling	Property defining how the LDAP should handle referrals.

Additional LDAP Resources

- http://www.openldap.org/
  This site contains the OpenLDAP main page. This site contains introduction, downloads, and documentation.
- http://www.iit.edu/~gawojar/ldap/
  This site is the LDAP browser site.
- http://ldap.akbkhome.com/
  This site contains an LDAP schema view with some definitions of the standard LDAP object classes and attributes.

9

Technical Design

This chapter contains information related to the technical design of ReIM.

Locking Design Summary

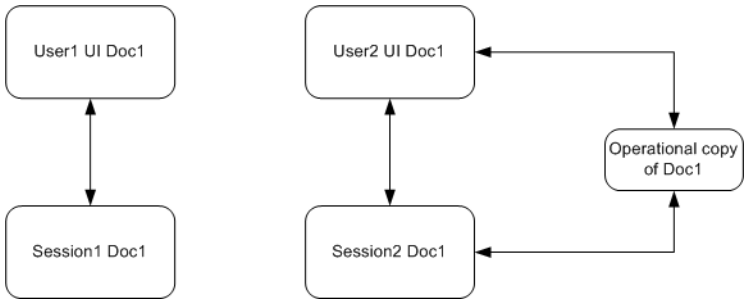
ReIM user interface is based on the optimistic locking strategy. It means that no entities will be preemptively locked before modification. Only at the point when modification needs to be performed the locking mechanism will be used. If two users are trying to modify the same entity, for example the same invoice, both users can do this on the screen. At the same time only the first user to attempt to save the changes will actually successfully save them and the second user will get an error message and would need to refresh the entity and try again.

Example

User1 is trying to update document D0. Document D0 has terms T0. User1 is trying to update T0 to T1. User2 is also trying to update document D0. User2 is trying to update T0 to T2. If User1 will be the first to attempt to save the change to D0 then D0 will have terms T1. User2 would need to refresh D0 to see the ne value of the terms and only after that would be able to change T1 to T2.

All user interface workflows can be sub-divided into 2 categories - simple entity workflows and complex process workflows. For simple entity workflows the locking is handled by the ADF framework. For complex workflows the workflow entities are copied into session (workspace) tables. For such workflows the locking is done by custom logic to allow data to be modified both by ADF framework code and by backend PL/SQL.

To facilitate locking, all tables that hold modifiable data have a standard - OBJECT_VERSION_ID - that is updated every time the entity is modified. New entity is created with OBJECT_VERSION_ID of 1. Modification verification is performed by comparing the copy's being modified version id with the original copy version id. If it different then the user's copy of the entity is out of sync.



In a batch mode no locking is performed. It is assumed that the batches will be run during the batch windows. It means the assumption is that he batch user will be the sole user of the entities. Because of this the batches should not be run ad-hoc during the normal business hours.

Currency Design Summary

ReIM has been designed to handle a multiple number of currencies. This section addresses the system's assumptions, conversion process, and validations that are related to this capability.

Merchandising System (such as RMS) and ReIM Assumptions

Consider the following assumptions.

- RMS defines one currency as the primary currency of the system (held on the RMS SYSTEM_OPTIONS table in the CURRENCY_CODE field).
- RMS specifies that each purchase order can have one currency. This purchase order currency does not have to be the same as the RMS primary system currency or the RMS supplier currency.
- ReIM requires that each document have its currency stated (IM_DOC_ HEAD.CURRENCY_CODE). This invoice currency does not have to be the same as the system primary currency.
- ReIM assumes that a purchase order and any invoices associated with that purchase order are in the same currency. This assumption is based on the business reality that these currencies are almost always the same and on the development consideration that currency conversion processes have an adverse impact on system performance.

Currency Conversion Process for Amount Tolerances

The following is information about the currency conversion process for amount tolerances.

- Amount tolerances are established in the currency of the tolerance entity (each tolerance entity has a currency setting). However, because the invoices and POs to be matched could reflect a different currency, amount tolerances must be converted before they can be applied. In other words, the currency established for amount tolerances is converted when the invoice/PO combination is not in the tolerance currency. For example, a tolerance defined as 10 US dollars (USD) has a much different meaning than a purchase order/invoice defined in Thai Baht (10 Thai Baht is about 0.23 USD). If the system merely utilized the number 10 and failed to perform a currency conversion, the amount tolerances would not apply correctly.
- Currency conversion rates are stored on the RMS CURRENCY_RATES table. The conversion factors on this table are in terms of the primary currency of the system. For example, suppose a retailer wishes to convert from Thai Baht to Uruguayan Pesos and the system's primary currency is USD. First, the system performs a conversion from Thai Baht to USD. Secondly, the system converts the USD value to Uruguayan Pesos. In other words, to perform its conversions, the system always must 'go through' the primary currency of the system.
- Currency conversion is always done based on he consolidated exchange rate.

Currency-Related System Validations

One of the validations performed by the EDI Injector process is that it determines whether the currency on the invoice is the same as the currency on the purchase order. If the invoice currency is not the same as the purchase order currency, the invoice is rejected.

The graphical user interface (GUI) invoice entry process also validates that the currency on the invoice is the same as the currency on the PO associated with the invoice. If the currencies are not the same, the user receives a warning message.

Currency Formatting

Monetary values must be properly formatted according to the associated currency for matting rule. The precision for the currency formatting is defined in RMS CURRENCIES table.

10

Oracle E-Business Suite Financials Integration using Oracle Retail Financial Integration

This chapter describes the integration between Oracle Retail systems and Oracle E-Business Suite Financials (including Oracle General Ledger and Oracle Payables), as developed and supported by Oracle Retail Financial Integration (ORFI).

When the option to integrate is chosen, the selected information is shared between the systems. Integration and validation services are in place to ensure the shared data matches.

Note: This chapter addresses the points within Oracle Retail systems that are essential to integration. For more information about the entire integration process, including mapping to Oracle E-Business Suite data and settings, see the ORFI document, *Oracle Retail Financial Integration for Oracle Retail Merchandising Suite and Oracle E-Business Suite Financials - Implementation Guide*. For more information about Web services, see the following chapters in the *Oracle Retail Merchandising System Operations Guide, Volume 2:* "Service Provider Implementations API Designs" and "Web Services."

Participating Applications

The following Oracle Retail applications are included in the integration covered by this chapter:

- Oracle Retail Merchandising System (RMS)
- Oracle Retail Sales Audit (ReSA)
- Oracle Retail Invoice Matching (ReIM)
- Oracle Retail Integration Bus (RIB)

Assumptions and Dependencies

- The option to integrate must be selected during initial set up of the RMS system.
- ReIM accesses RMS to determine if integration is active. The RMS set up must be done before integrating with REIM.

- The URLs for the RFI Web services that are necessary for this integration are maintained in the RETAIL_SERVICE_REPORT_URL table and in the ReIM IM_SYSTEM_OPTIONS table.
- Real time account validation is done only when the financial integration with Oracle E-Business Suite is ON.
- Partners must be set up as suppliers in Oracle E-Business Suite. Then the partner must be manually set up in RMS using the RMS Supplier ID that was generated when the Oracle E-Business Suite supplier was interfaced to Oracle Retail. Partner functionality within RMS and ReIM can then proceed normally. The RMS supplier generated as part of this process is not used.
- Payment terms and freight terms are manually maintained.

Data Setup

Integration of Oracle Retail applications and Oracle E-Business Suite Financials relies on synchronization of essential data, such as currency exchange rates and suppliers. Through careful discussions, the users of both systems determine the common codes and descriptions that will best serve their business needs.

When an agreement is reached, this information is set up and maintained. Depending on the volume, some shared information is set up in Oracle Retail applications or in Oracle E-Business Suite and electronically transferred to other systems. Otherwise, shared information is set up manually within each system, and the users of both systems must ensure that the code and the description match.

RMS Data Setup and Configuration

This section describes setup considerations for the RMS data.

RMS System Options

As part of the RMS system options setup, set the following options as indicated:

This SYSTEM_OPTION indicates that the Oracle Retail system is integrated with a financial system:

- FINANCIAL_AP=A

A value of A indicates that the financial system to which RMS is interfaced is Oracle E-Business Suite through Oracle Retail Financial Integration (ORFI).

- GL_ROLL_UP can be D/S/C
- SUPPLIER_SITE_IND = Y
- ORG_UNIT_IND = Y

Organization Units

Use the Organizational Unit window (RMS Start Menu > Control > Setup > Org Unit >Edit) to define organizational units in RMS that match those being setup in Oracle E-Business Suite. When an organizational unit is entered in RMS, the valid organizational units are those associated with the Set Of Books (SOB) used for the general ledger interface.

Currency Exchange Rates

Currency exchange rate is used to translate the monetary value of one currency in terms of another. Depending on business needs, a Currency Exchange Rate Type of Operational or Consolidation is selected for use in all transactions.

This value is set up manually in RMS and mapped to Oracle E-Business Suite through the Currency Exchange Type mapping window. Currency Exchange Rate data is owned by Oracle General Ledger and updates are sent to Oracle Retail applications.

Determine the Exchange Type being sent by Oracle General Ledger (for example, Consolidation or Operational) that you want RMS to use.

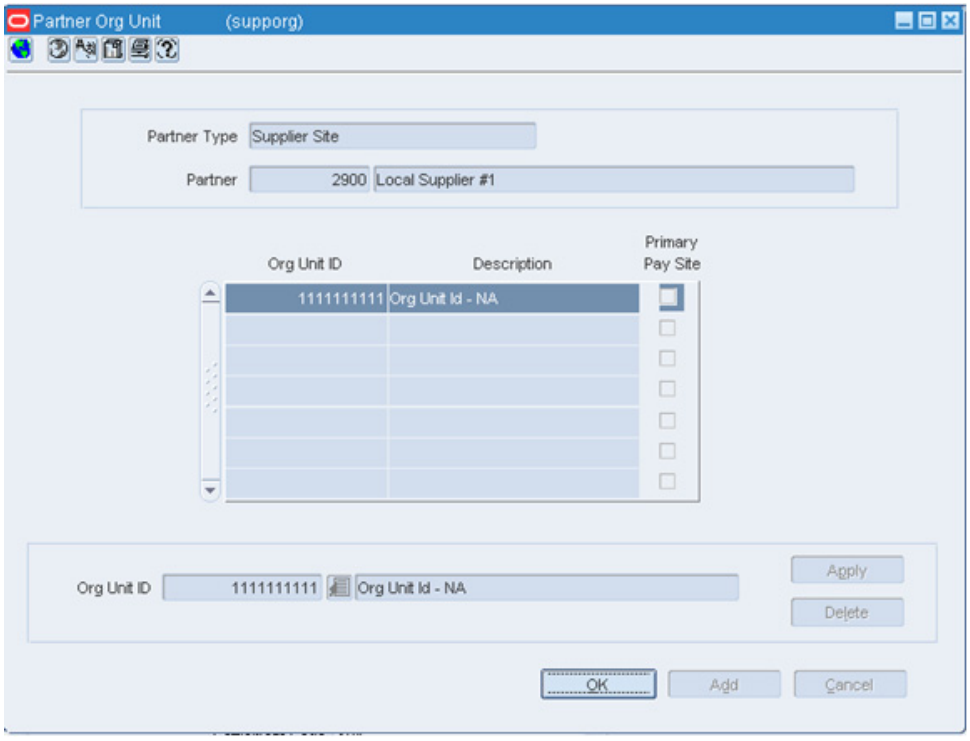
Update the FIF_CURRENCY_XREF for mapping the external exchange type being sent by Oracle General Ledger with RMS Exchange Type. For example, for Consolidation and Operational exchange types, the FIF_CURRENCY_XREF table holds the following entries:

Table 10–1

FIF_EXCHANGE_TYPE	RMS_EXCHANGE_TYPE
C	C
O	O

Supplier Address Types

Within RMS, supplier information (such as Order From and Remit To addresses) is used for generating the purchase orders. Oracle Payables uses supplier information for payment generation. It is important that this information is synchronized.



Suppliers are created in Oracle Payables and exported to RMS. When FINANCIAL_AP is set to A, suppliers cannot be created using the RMS forms. However, if a supplier exists in RMS, all data values for the supplier (except supplier name and status) are

updated using the RMS forms. The association of supplier sites to organization units is accessed only in view mode through RMS forms. One supplier site per supplier and organization unit combination can be marked as primary payment site.

Where SYSTEM_OPTIONS.FINANCIAL_AP is A, disable auto generate supplier/partner numbers and associated check boxes.

> **Note:** Supplier information is created, updated and inactivated only in Oracle Payables. This information is transferred from Oracle Payables to the participating Oracle Retail applications, where additional retail-specific attributes are maintained.

Country Codes

When country codes are defined and seeded in RMS, ensure that country codes are mapped to Oracle E-Business Suite country codes through RFI DVM mapping. The following is an example of RFI DVM Mapping (Table RFI_XREF_DVM, available in RFI schema in Retail.

Table 10–2

EXT_SYSTEM_ID	COMMON_ID	RETL_ID
USA	700	US
CAN	701	CA

Financial Calendar

The financial calendar within Oracle Retail systems is manually set up and maintained separately from the Oracle General Ledger financial calendar.

Freight Terms

A freight term is an agreement between the retailer and a supplier regarding transportation charges for goods delivered by the supplier. Freight terms are used by RMS as purchase orders are generated.

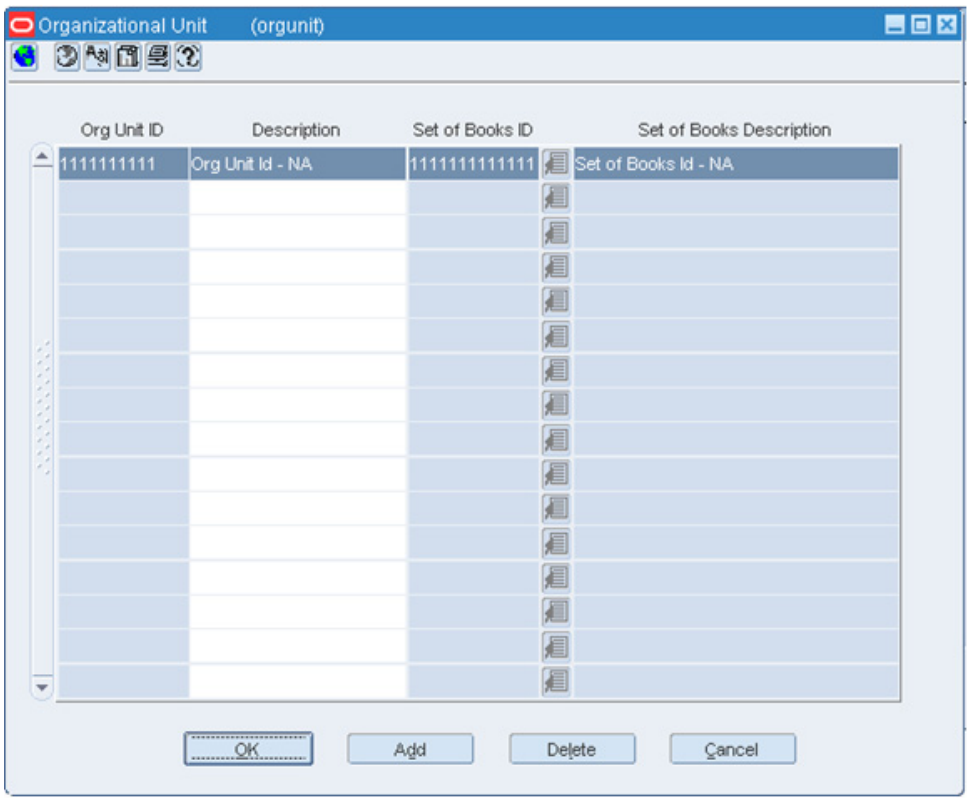
Within the RMS system, freight terms are set up and maintained manually. They are also maintained in Oracle Payables.

Payment Terms and Currency Exchange Rates

Currency, exchange rates are created and updated in Oracle General Ledger and exported to RMS. Changes to Retail currency exchange rates are not propagated to Oracle General Ledger. Payment terms, however, are manually set up and maintained in each system.

Oracle E-Business Suite Financials Units and Site IDs

The data concepts of Org Units and Site IDs in RMS mirror the data maintained in Oracle E-Business Suite. RMS forms are used to manage and view Oracle Org Units and Site IDs. The RMS windows for Store and Warehouse maintenance allow for the association of each store and warehouse with an Org Unit. The following is an example of the Organizational Unit form:



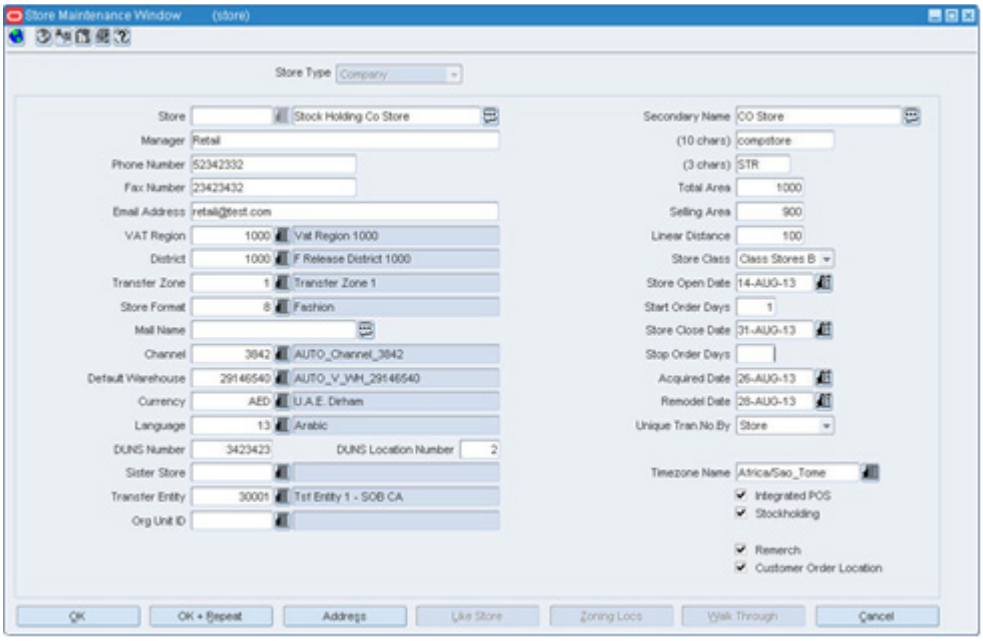
Store and Warehouse Maintenance

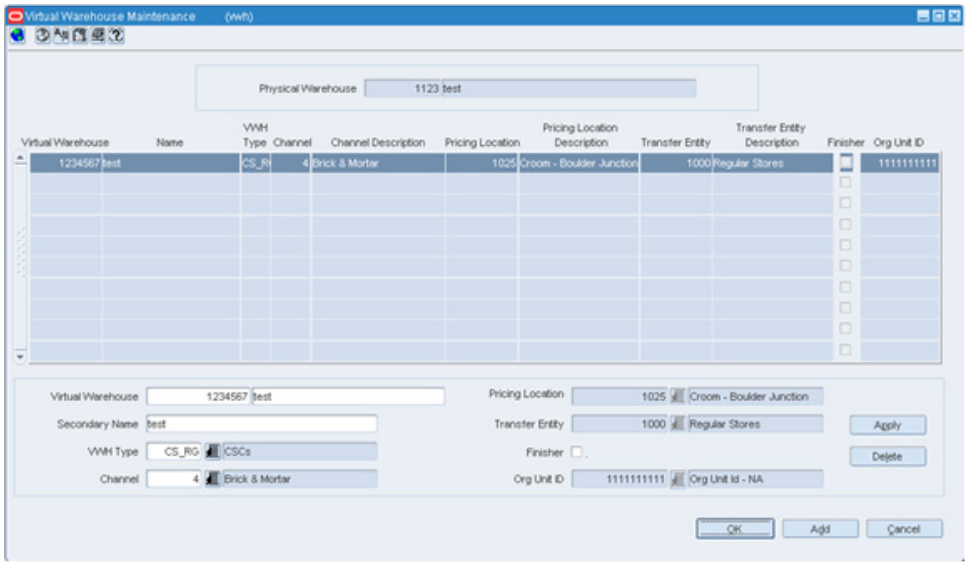
The organizational unit is found in the Store Maintenance and Warehouse forms, which allow the Oracle E-Business Suite operating unit to be associated with the Store or Warehouse.

- When RMS is set up for multi-channel operation, the organizational unit is set up at the virtual warehouse level.

Financial sales audit and inventory information is identified through interface routines and posted to the appropriate general ledger accounts. An organizational unit must be designated for each Store and Warehouse location in the RMS.

The following are examples of the Store Maintenance and Warehouse Maintenance forms:





RMS General Ledger Setup

For RMS and ReSA, manual setup is required for validating the chart of accounts. Valid chart of accounts are created and stored in general ledger cross reference tables. Once the validation is completed, transaction data is assigned to specific account codes.

Ongoing maintenance of the chart of accounts information (such as adding, changing, or deleting chart of accounts) requires re-validation. In this regard, Oracle General Ledger is the system of record, as it is used to verify the chart of accounts used by Oracle Retail applications. When these applications send a chart of accounts for validation, Oracle General Ledgers issues a message with:

- Valid or invalid status
- Response date

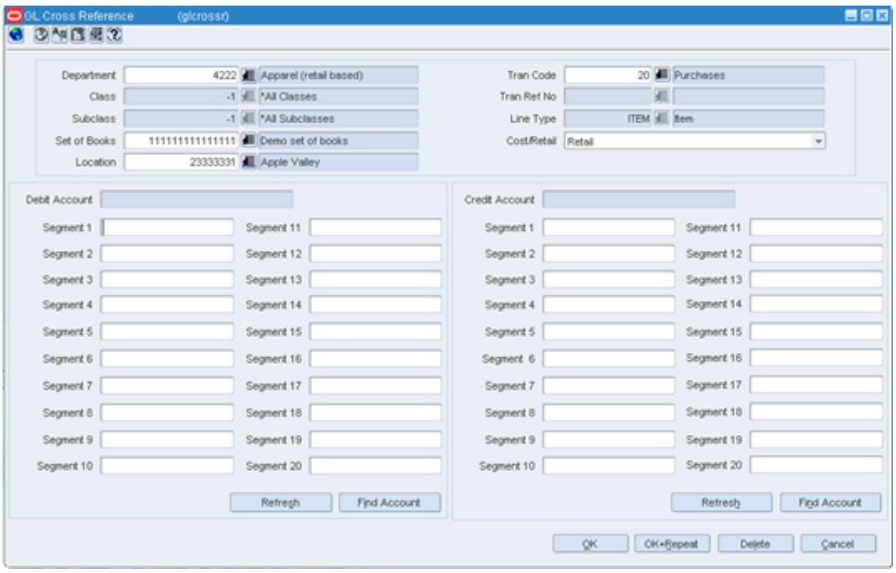
- Chart of accounts

The RMS table FIF_GL_SETUP, stores the Oracle E-Business Suite Set of Books IDs to post financial information. This table must be setup manually after Set of Books IDs are determined. Where a system indicator Multiple Set of Books ID is set to N, FIF_GL_SETUP must hold a single Set of Books (SOB) record.

The Set of Books IDs is associated with the chart of accounts when setting up general ledger cross-reference.

RMS General Ledger Cross Reference

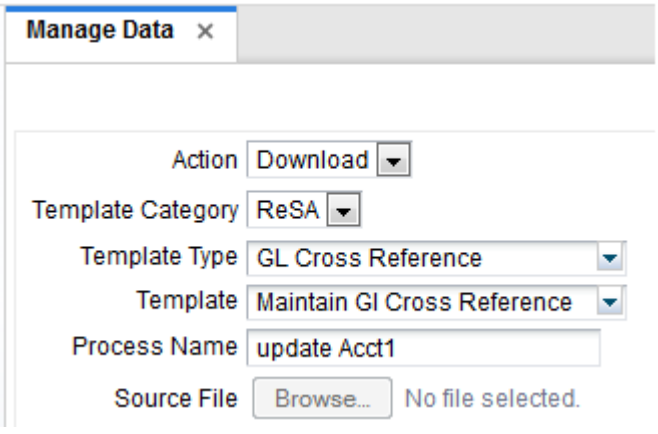
Select **RMS Start Menu > Finance> GL Cross Reference**. The General Ledger Search window opens. Map Chart of Accounts to department, Class, Subclass, Set Of Books, location, and transaction codes using the GL cross reference form in RMS.



ReSA General Ledger Cross Reference

Select **ReSA task list > Foundation Data > Data Loading > Manage Data**. The manage data screen opens. Through this screen, the user can choose the General Ledger template to maintain the data using spreadsheet download and upload.

When SYSTEM_OPTIONS.FINANCIAL_AP is A, the upload validates entries of valid segment combinations.



ReIM Data Setup and Configuration

This section describes setup considerations for ReIM data.

System Options

As part of the RMS system options setup script, set the following options as indicated:

- FINANCIAL_AP =A

Chart of Accounts Setup

The chart of accounts is set up manually in Oracle Retail applications and in Oracle General Ledger. All account combinations are set up in each Set Of Books. Account validation is done while executing Financial Posting batch:

Segment Mapping

The retailer determines how many segments are populated. Up to 20 account segments can be specified. The following is an example of how segments are mapped between the ReIM transaction table and Oracle General Ledger:

Table 10–3

ReIM Segments	Oracle General Ledger Chart of Accounts
Segment 1	PRODUCT
Segment 2	ACCOUNT
Segment 3	ALTACCT
Segment 4	OPERATING_UNIT
Segment 5	FUND_CODE
Segment 6	DEPTID
Segment 7	PROGRAM_CODE
Segment 8	CLASS_FLD
Segment 9	BUDGET_REF
Segment 10	BUSINESS_UNIT_PC
Segment 11	PROJECT_ID
Segment 12	ACTIVITY_ID
Segment 13	RESOURCE_TYPE
Segment 14	RESOURCE_CATEGORY
Segment 15	RESOURCE_SUB_CAT
Segment 16	CHARTFIELD1
Segment 17	CHARTFIELD2
Segment 18	CHARTFIELD3
Segment 19	AFFILIATE
Segment 20	AFFILIATE_INTRA1

If any one of the values in the 20 segments does not match the Oracle General Ledger, the account combination is considered as invalid. The following error message is

added to IM_POSTING_DOC_ERRORS table; "One or more accounts subjected to posting is invalid".

Segments 1 and 2 may be set up as dynamic at the Location level, or Segments 4 and 5 can be dynamic at the Department and Class level respectively. Segments defined as dynamic are allowed to be null for certain types of Basic Transaction or Reason Code cross-reference types. When a segment is null, the segment is assigned dynamically when transactions are posted. (Non-dynamic segments cannot be blank). Validation applies to the segment combination, not to individual segments.

Note: For Tran code TAP, each segment must have a value regardless of whether the segment is dynamic.

Running the Initial Load from Oracle E-Business Suite Financials

The initial load for ReIM is run by Oracle E-Business Suite and includes the following information:

- Suppliers
- Currency Rates

Note: The view, mv_currency_conversion_rates should be refreshed once the initial loads of currencies from Oracle General Ledger are loaded to ReIM.

IM_SYSTEM_OPTIONS Table Setup

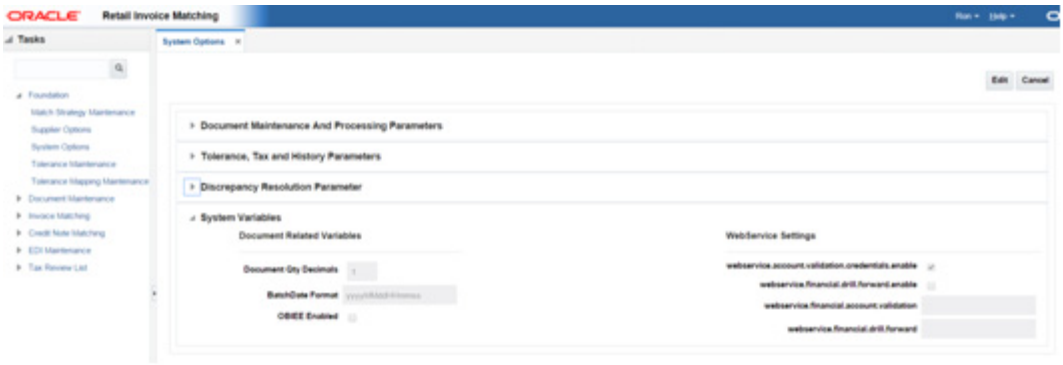
To accommodate integration, the IM_SYSTEM_OPTOINS table should be configured with the following properties.

WS_FIN_ACC_VALID_URL - This attribute holds the URL for the financial account validation web service.
WS_FIN_ACC_VALID_CRED - This attribute indicates if the account validation web service call is authenticated or not. Y means Authentication enabled. N means Authentication is not enabled (See configuring web service credentials in weblogic server).

WS_FIN_DRILL_FWD_URL - This attribute holds the URL for the financial drill forward web service.

WS_FIN_DRILL_FWD_CRED - This attribute indicates if the drill forward web service call is authenticated or not. Y means Authentication enabled. N means Authentication is not enabled (See configuring web service credentials in weblogic server).

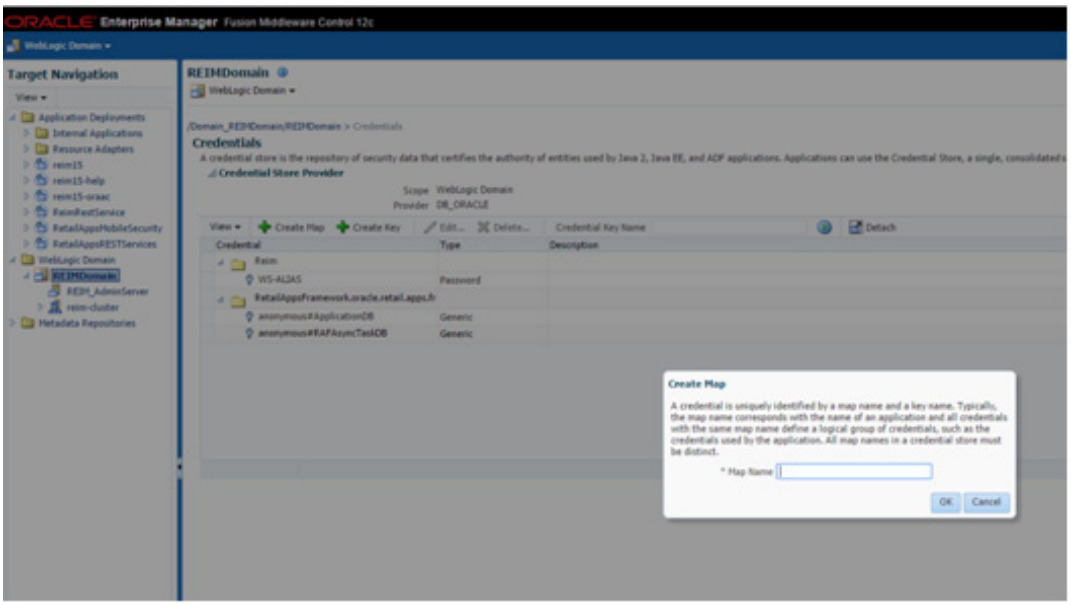
The above attributes are configured in the System Options screen.



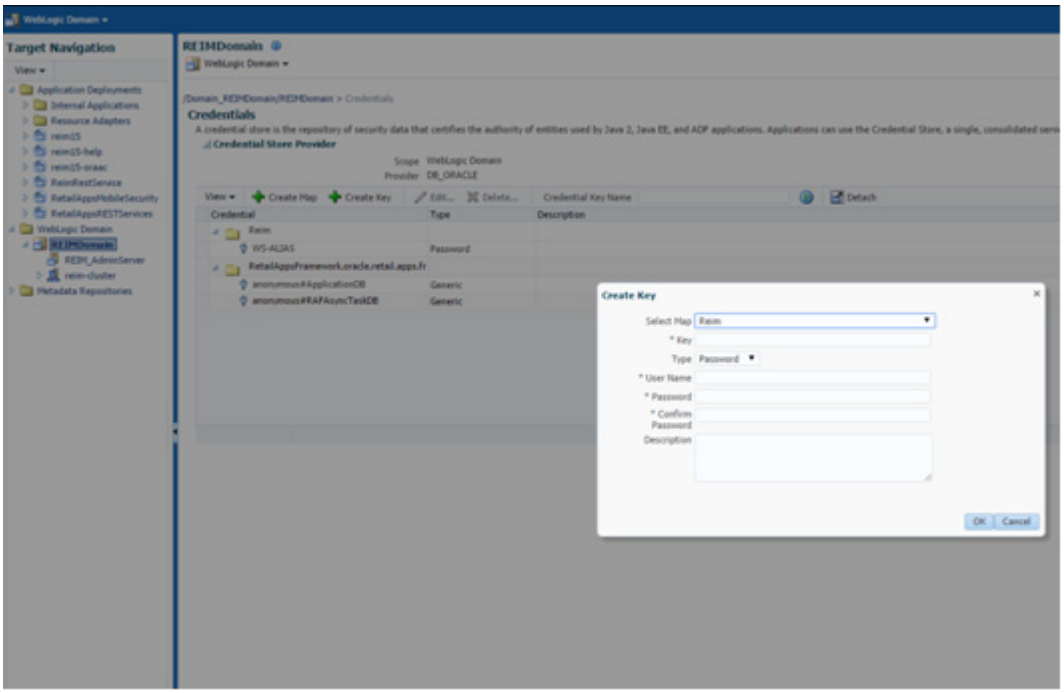
Configuring WebService Credentials in Weblogic Server Enterprise Manager

The credentials for the web service call are configured in the default domain credential store of the WebLogic server through the admin console.

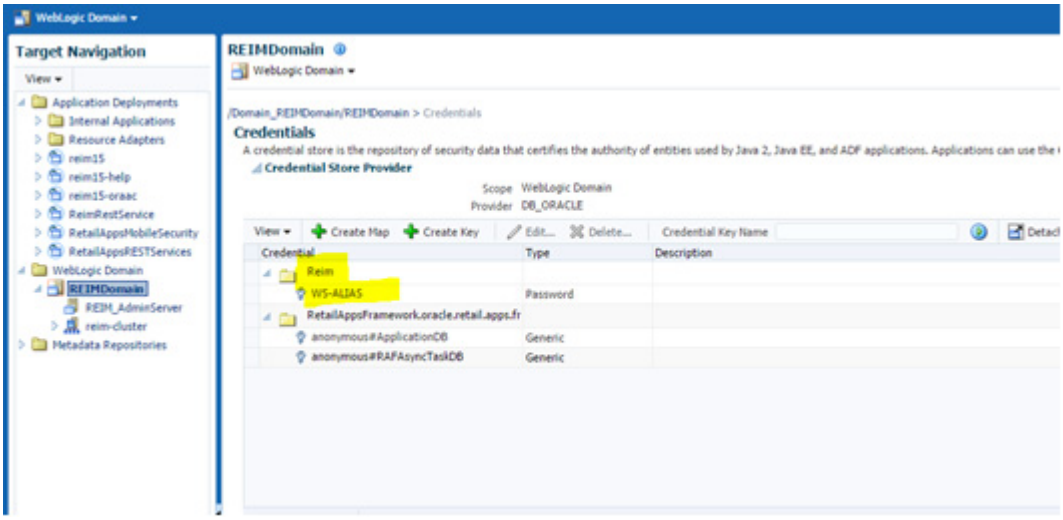
Create Map:



Creating Credentials:



Select the map as Reim. The Key name should always be WS-ALIAS. Any user name / password can be assigned to the key.



Reports are created by Business Intelligence Publisher for the following:

Table 10–4

Document Type	Report Name	Sample Report URL
MRCHI	Merchandise invoice document Report	http://hostname:portno/xmlpserver_nonsso/Guest/REIM13/Finance/invreport/invreport.xdo
NMRCHI	Non-Merchandise invoice document Report	http://hostname:portno /xmlpserver_nonsso/Guest/REIM13/Finance/invreport/invreport.xdo
CRDNT	Credit Note document Report	http://hostname:portno/xmlpserver_nonsso/Guest/REIM13/Finance/invreport/crnreport.xdo

Table 10–4 (Cont.)

Document Type	Report Name	Sample Report URL
CRDMEC	Credit Memo cost document Report	http://hostname:portno/xmlpserver_nonsso/Guest/REIM13/Finance/invreport/memoreport.xdo
CRDMEQ	Credit Memo quantity document Report	http://hostname:portno/xmlpserver_nonsso/Guest/REIM13/Finance/invreport/memoreport.xdo
DEBMEC	Debit Memo cost document Report	http://hostname:portno/xmlpserver_nonsso/Guest/REIM13/Finance/invreport/imemoreport.xdo
DEBMEQ	Debit Memo quantity document Report	http://hostname:portno/xmlpserver_nonsso/Guest/REIM13/Finance/invreport/memoreport.xdo
DEBMEV	Debit Memo Tax document Report	http://hostname:portno/xmlpserver_nonsso/Guest/REIM13/Finance/invreport/memoreport.xdo
RWO	Receipt Write Off document Report	http://hostname:portno/xmlpserver_nonsso/Guest/REIM13/Finance/invreport/rworeport.xdo

The URL for each report must be updated in the table, RETAIL_SERVICE_REPORT_URL. The following table provides sample URLs.

ReIM Transactional Maintenance

Integration to Oracle General Ledger includes a number of transactions, as described below.

Calculation of TRANS_AMOUNT

The TRANS_AMOUNT field in the IM_FINANCIAL_STAGE table stores the value of the journal entry to be posted to Oracle General Ledger. (The currency for the calculated amount is the currency assigned to the transaction). The TRANS_AMOUNT value is calculated as follows:

Table 10–5

Row Description	DEBIT_CREDIT_IND	TRANS_AMOUNT Value
Normal	Debit	Transaction Amount
Normal	Credit	(-1) * Transaction Amount
VAT	Debit	Transaction Amount * VAT Rate
VAT	Credit	(-1) * Transaction Amount * VAT Rate

Note: Transaction Amount is taken from the database column, IM_FINANCIALS_STAGE.AMOUNT.

Generation of Outgoing Data

A staging table accommodates the outgoing transfer of data. The reference key assigned to each document or receipt is used to find data on this table.

Table 10–6

From	To	Transactions
ReIM	Oracle Payables	■ Invoices ■ Debit Memos ■ Credit Memos ■ Credit Notes
ReIM	Oracle General Ledger	General Ledger accounting entries resulting from the Invoice Matching process, including: ■ Pre-paid invoices ■ Receipt Write-offs
RMS	Oracle General Ledger	Accounting entry data (potentially very high volume)
ReSA	Oracle General Ledger	Accounting entry data (potentially very high volume)

Validation of Accounts When Posting Financial Entries

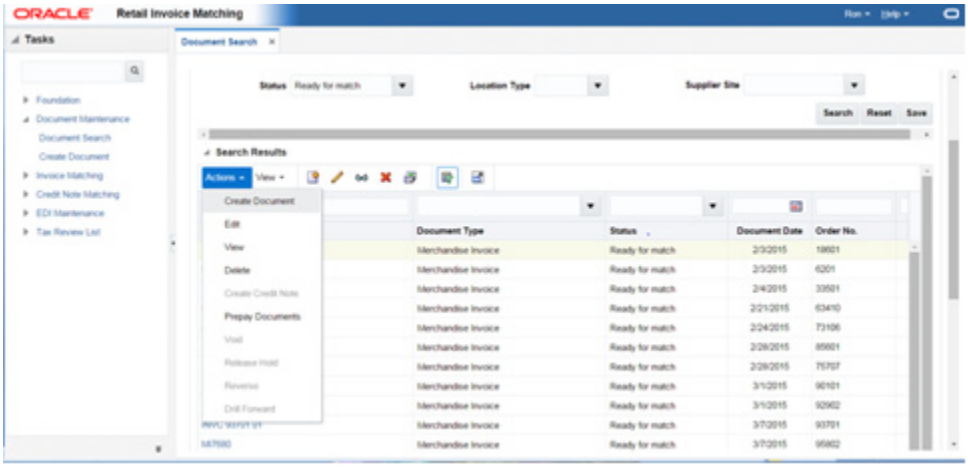
Valid chart of accounts are stored in the ReIM table, IM_VALID_ACCOUNTS, which includes the Set of Books ID (sob_id) and 20 segments. An ORFI Web service validates accounts against the Oracle General Ledger. Valid accounts are posted to IM_VALID_ACCOUNTS; invalid accounts are posted to IM_POSTING_DOC_ERROR. The following steps describe the validation process:

1. The ReIM system invokes the validation Web service to validate the chart of accounts. (A URL for the ORFI Web service is configured in the SYSTEM_OPTIONS table
2. The posting batch job checks the accounts to be posted against the IM_VALID_ACCOUNTS table.
3. If the chart of accounts is in the table, the transaction is posted to staging tables.
4. If the chart of account does not exist in the table, a collection of accounts is built. These collected accounts are validated against the Oracle General Ledger, and a status is returned.
    - If the status of the collected accounts is valid, the accounts are inserted in the IM_VALID_ACCOUNTS table, and the transactions are posted to the staging tables.
    - If the status of the accounts is NOT valid, the entire collection is flagged as errors, and transactions are posted to IM_POSTING_DOC_ERROR.

Note: ReIM completes the first level of account validation and posts the transaction to staging tables. It is assumed the second level of account validation is done at the end of the extraction process (where transactions are moved from ReIM staging tables to Oracle General Ledger). If account validation fails at this point, Oracle General Ledger must change the account information before transactions are loaded to Oracle General Ledger, and the chart of accounts must be re-validated in ReIM.

Validation of Accounts When Prepaying a Merchandise Invoice

A Merchandise document with a ready for match status can be pre paid before matching. The Document Search screen provides this option.



Valid chart of accounts are stored in the ReIM table, IM_VALID_ACCOUNTS, which includes the Set of Books ID (sob_id) and 20 segments. An ORFI Web service validates accounts against the Oracle General Ledger. Valid accounts are posted to IM_VALID_ACCOUNTS; invalid accounts are posted to IM_POSTING_DOC_ERROR. The following steps describe the validation process:

1. The ReIM system invokes the validation Web service to validate the chart of accounts. (A URL for the ORFI Web service is configured in the SYSTEM_OPTIONS table.
2. The Prepay operation checks the accounts to be posted against the IM_VALID_ACCOUNTS table.

   If the chart of accounts is in the table, the transaction is posted to staging tables.

   If the chart of account does not exist in the table, a collection of accounts is built. These collected accounts are validated against the Oracle General Ledger, and a status is returned.

   - If the status of the collected accounts is valid, the accounts are inserted in the IM_VALID_ACCOUNTS table, and the transactions are posted to the staging tables.
   - If the status of the accounts is NOT valid, the entire collection is flagged as errors, and transactions are posted to IM_POSTING_DOC_ERROR and the user will be shown an error message "One or more accounts subjected to posting is invalid".

Maintenance of Valid Accounts

As account information is changed in the Oracle General Ledger, Retail must re-validate the locally stored chart of accounts. Oracle General Ledger will not propagate chart of account changes to Retail. The AccountWorkspacePurge Batch can clear all valid accounts in the IM_VALID_ACCOUNTS table or only those that are considered updates in Oracle E-Business Suite.

Usage

```
AccountWorkspacePurge userid/password PURGE [ALL | <Accounts>]
```

Where:

- The argument is a combination of user ID and password.
- The argument is the word PURGE.
- The argument is either ALL or specific accounts to be deleted from the local table.

11

PeopleSoft Financials Integration using Oracle Retail Financial Integration

This chapter describes the integration between Oracle Retail systems and PeopleSoft Financials (including PeopleSoft General Ledger and PeopleSoft Payables), as developed and supported by Oracle Retail Financial Integration (ORFI).

When the option to integrate is chosen, the selected information is shared between the systems. Integration and validation services are in place to ensure the shared data matches.

Note: This chapter addresses the points within Oracle Retail systems that are essential to integration. For more information about the entire integration process, including mapping to data and settings, see the ORFI document, *Oracle Retail Financial Integration for Oracle Retail Merchandise Operations Management and Oracle Financials Implementation Guide*. For more information about Web services, see the following chapters in the *Oracle Retail Merchandising System Operations Guide, Volume 2:* "Service Provider Implementations API Designs" and "Web Services."

Participating Applications

The following Oracle Retail applications are included in the integration covered by this chapter:

- Oracle Retail Merchandising System (RMS)
- Oracle Retail Sales Audit (ReSA)
- Oracle Retail Invoice Matching (ReIM)
- Oracle Retail Integration Bus (RIB)

Assumptions and Dependencies

- The option to integrate must be selected during initial set up of the RMS system.
- ReIM accesses RMS to determine if integration is active. Initial set up of RMS must occur prior to the integration of ReIM.
- The URLs for the RFI Web services that are a necessary for this integration must be maintained in the RETAIL_SERVICE_REPORT_URL table and in the ReIM SYSTEM_OPTIONS table.

- Real time account validation is done only when the financial integration with PeopleSoft Financials is ON.
- Partners must be set up as suppliers in PeopleSoft. Then the partner must be manually set up in RMS using the RMS Supplier ID that was generated when the PeopleSoft supplier was interfaced to Oracle Retail. Partner functionality within RMS and ReIM can then proceed normally. The RMS supplier generated as part of this process is not used.
- Payment terms and freight terms are manually maintained.

Data Constraints

- The Location ID field is restricted to eight characters, to accommodate PeopleSoft Operating Unit, which has a maximum of eight characters.
- The Ext_Doc_ID field is restricted to 30 characters, because the corresponding PeopleSoft field has only 30 characters. Characters beyond 30 are truncated.
- RMS allows for four decimals, and PeopleSoft allows only three. Truncation may occur when data is passed to PeopleSoft Enterprise Financials.
- ReIM values in the IM_CURRENCY_LOCALE are restricted to three decimals, because the corresponding PeopleSoft Enterprise Financials field can accept no more than three decimal positions.

Data Setup

Integration of Oracle Retail applications and PeopleSoft Financials relies on synchronization of essential data, such as currency exchange rates, suppliers, and payment terns. Through careful discussions, the users of both systems determine the common codes and descriptions that will best serve their business needs.

Once agreement is reached, this information is set up and maintained. Depending on the volume, some shared information is set up in Oracle Retail applications or in PeopleSoft and electronically transferred to the other systems. Otherwise, shared information is set up manually within each system, and the user of both systems must ensure that the code and the description match.

RMS Data Setup and Configuration

This section describes setup considerations for the RMS data.

This system_option indicates that the Oracle Retail system is integrated with a financial system:

- FINANCIAL_AP=A

A value of A indicates that the financial system to which RMS is interfaced is PeopleSoft through Oracle Retail Financial Integration (ORFI).

- GL_ROLL_UP can be D/S/C
- SUPPLIER_SITE_IND = Y
- ORG_UNIT_IND = Y

Organization Units

Use the Organizational Unit window (**RMS Start Menu > Control > Setup > Org Unit >Edit**) to define organizational units in RMS that match those being setup in

PeopleSoft. When an organizational unit is entered in RMS, the valid organizational units are those associated with the Set Of Books (SOB) that is being used for the general ledger interface.

Currency Exchange Rates

Currency exchange rate is used to translate the monetary value of one currency in terms of another. Depending on business needs, a Currency Exchange Rate Type of Operational or Consolidation is selected for use in all transactions.

This value is set up manually in RMS and mapped to PeopleSoft through the Currency Exchange Type mapping window. Currency Exchange Rate data is owned by PeopleSoft General Ledger and updates are sent to Oracle Retail applications.

Determine the Exchange Type being sent by PeopleSoft General Ledger (for example, Consolidation or Operational) that you want RMS to use. Update the FIF_CURRENCY_XREF for mapping the external exchange type being sent by PeopleSoft General Ledger with RMS Exchange Type.

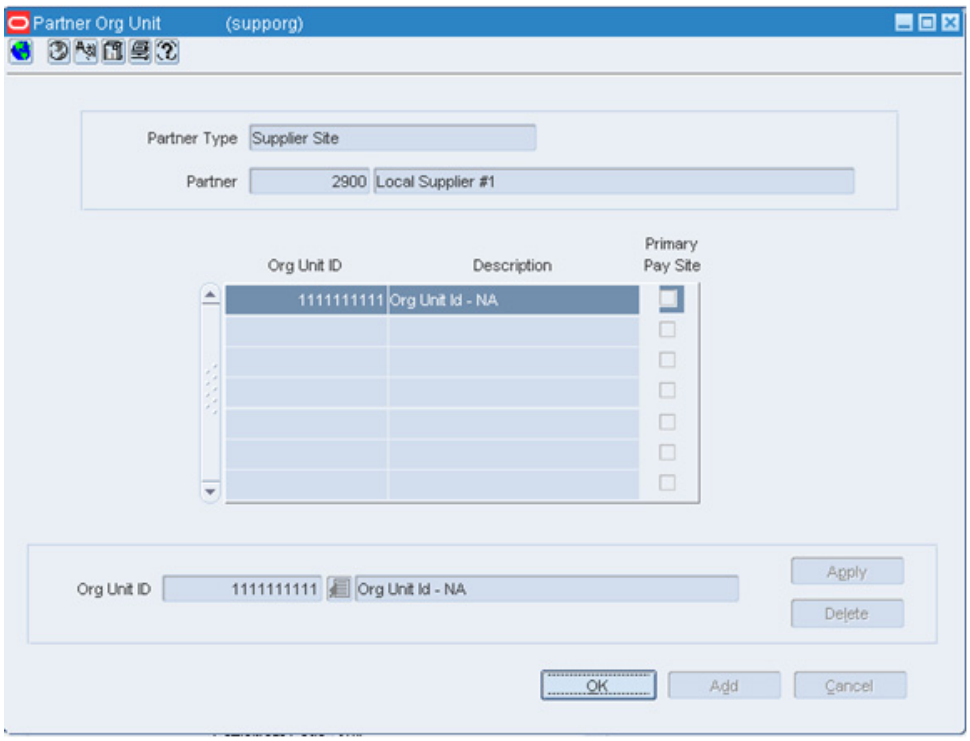
For example, for Consolidation and Operational exchange types, the FIF_CURRENCY_XREF table holds the following entries:

Table 11–1

FIF_EXCHANGE_TYPE	RMS_EXCHANGE_TYPE
C	C
O	O

Supplier Address Types

Within RMS, supplier information (such as Order From and Remit To addresses) is used for generating the purchase orders. PeopleSoft Payables uses supplier information for payment generation. It is important that this information is synchronized.



Suppliers are created in PeopleSoft Payables and exported to RMS. When FINANCIAL_AP is set to A, suppliers cannot be created using the RMS forms. However, after the supplier exists in RMS, all data values for the supplier (except supplier name and status) continue to be updated using the RMS forms. The association of supplier sites to organization units is accessed only in view mode through RMS forms. One supplier site per supplier organization unit combination can be marked as primary payment site.

Where SYSTEM_OPTIONS.FINANCIAL_AP is A, disable auto generate supplier/partner numbers and associated check boxes.

Note: Supplier information is created, updated and inactivated only in PeopleSoft Payables. This information is transferred from PeopleSoft Payables to the participating Oracle Retail applications, where additional retail-specific attributes may be maintained.

Country Codes

When country codes are defined and seeded in RMS, ensure that country codes are mapped to PeopleSoft country codes through RFI DVM mapping. The following is an example of RFI DVM Mapping (Table RFI_XREF_DVM, available in RFI schema in Retail.

Table 11–2

EXT_SYSTEM_ID	COMMON_ID	RETL_ID
USA	700	US
CAN	701	CA

Financial Calendar

The financial calendar within Oracle Retail systems is manually set up and maintained separately from the PeopleSoft General Ledger financial calendar..

Freight Terms

A freight term is an agreement between the retailer and a supplier regarding transportation charges for goods delivered by the supplier. Freight terms are used by RMS as purchase orders are generated.

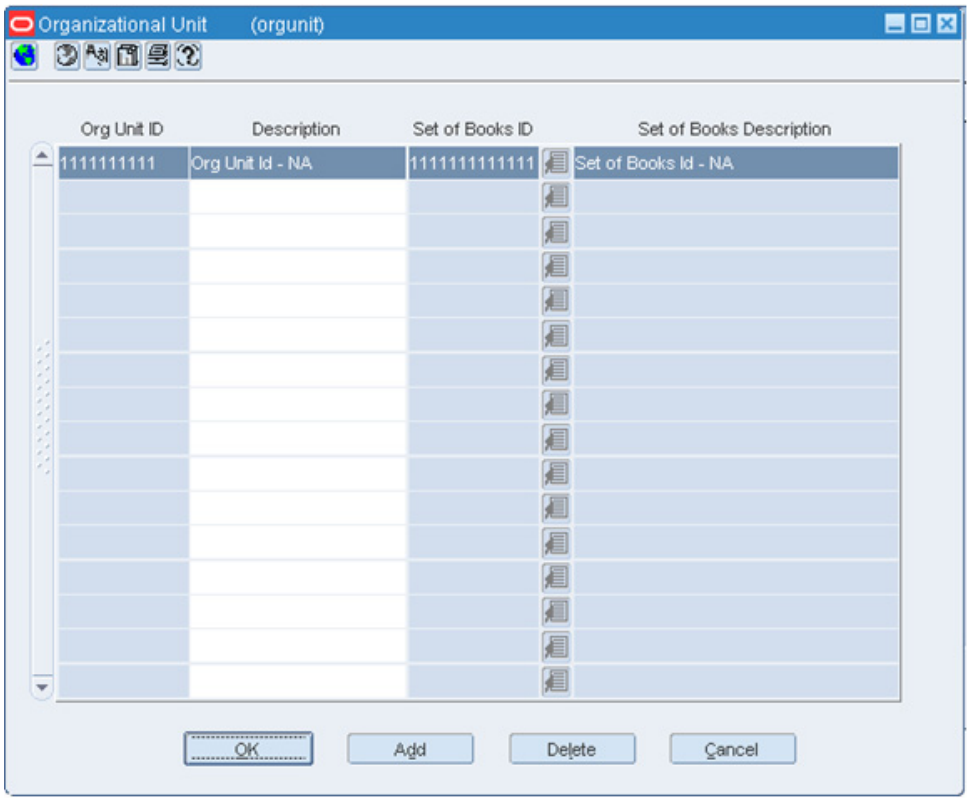
Within the RMS system, freight terms are set up and maintained manually. They also are maintained in PeopleSoft Payables.

Payment Terms and Currency Exchange Rates

Currency exchange rates are created and updated in PeopleSoft General Ledger and exported to RMS. Changes to Retail currency exchange rates are not propagated to PeopleSoft General Ledger. Payment terms set up in PeopleSoft are propagated to RMS but changes to payment terms in RMS are not propagated back to PeopleSoft.

PeopleSoft Financials Units and Site IDs

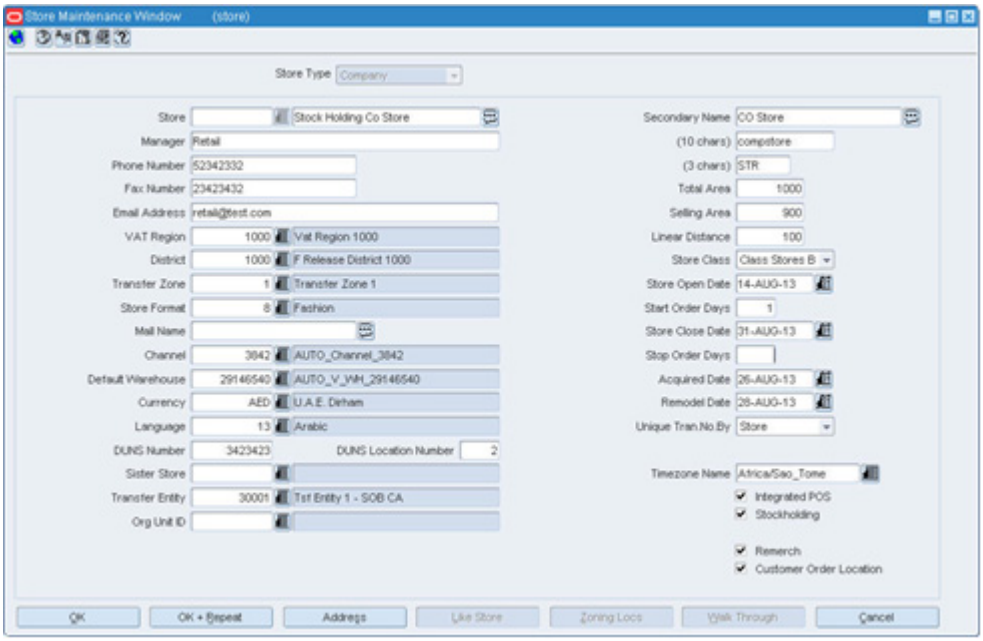
The data concepts of Org Units and Site IDs in RMS mirror the data maintained in PeopleSoft. RMS forms are used to manage and view Oracle Org Units and Site IDs. The RMS windows for Store and Warehouse maintenance allows for the association of each store and warehouse with an Org Unit. The following is an example of the Organizational Unit form:

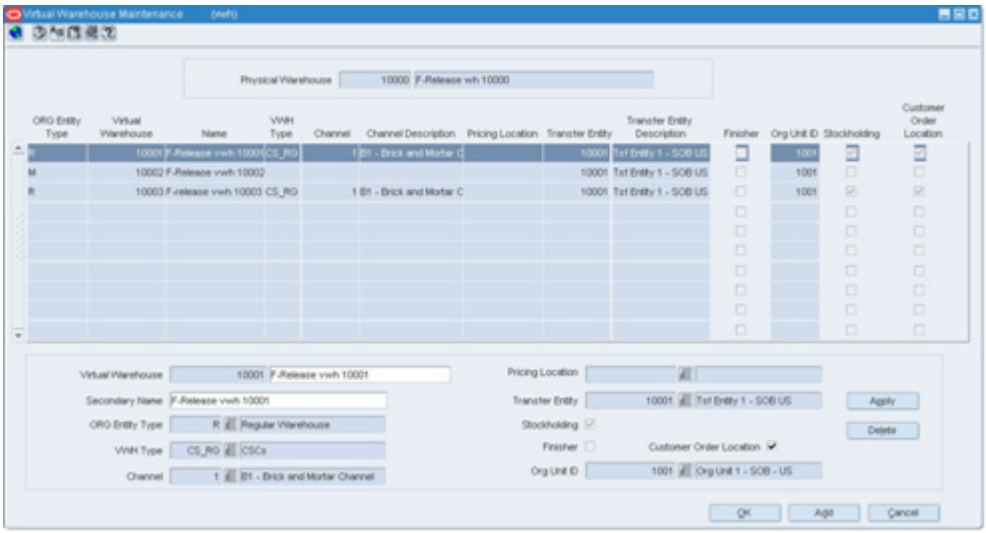


Store and Warehouse Maintenance

The organizational unit is found in the Store Maintenance and Warehouse forms, which allow the PeopleSoft operating unit to be associated with the Store or Warehouse. When RMS is set up for single-channel operation, the organizational unit is set at the physical warehouse level. When RMS is set up for multi-channel operation, the organizational unit is set up at the virtual warehouse level. Financial sales audit and inventory information can then be identified through interface routines and posted to the appropriate general ledger accounts. An organizational unit must be designated for each Store and Warehouse location in the RMS.

The following are examples of the Store Maintenance and Warehouse Maintenance forms:





RMS General Ledger Setup

For RMS and ReSA, manual setup is required for validating the chart of accounts. Valid chart of accounts are created and stored in general ledger cross reference tables.

Once the validation is completed, transaction data can be assigned to specific account codes.

Ongoing maintenance of the chart of accounts information (such as adding, changing, or deleting chart of accounts) requires re-validation. In this regard, PeopleSoft General Ledger is the system of record, as it is used to verify the chart of accounts used by Oracle Retail applications. When these applications send a chart of accounts for validation, PeopleSoft General Ledgers issues a message with:

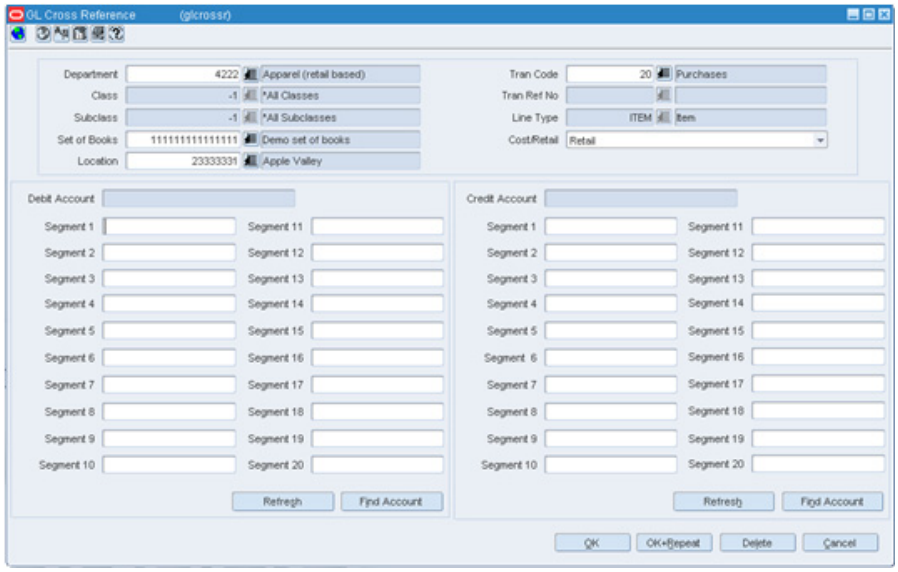
- Valid or invalid status
- Response date
- Chart of accounts

The RMS table, FIF_GL_SETUP, stores the PeopleSoft Set of Books IDs to post financial information. This table must be setup manually after Set of Books IDs are determined. Where system indicator Multiple Set of Books ID is set to N, FIF_GL_SETUP must hold a single Set of Books (SOB) record.

The Set of Books IDs is associated with the chart of accounts when setting up general ledger cross reference.

RMS General Ledger Cross Reference

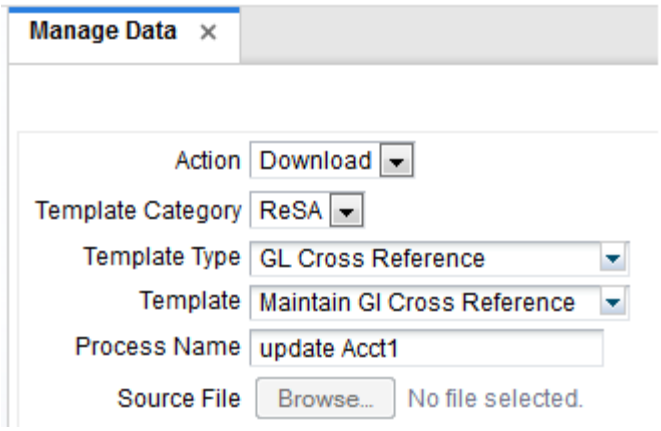
Navigate: RMS Start Menu > Finance> GL Cross Reference. The General Ledger Search window opens. Map Chart of Accounts to department, Class, Subclass, Set Of Books, location, and transaction codes using the GL cross reference form in RMS.



ReSA General Ledger Cross Reference

Navigate: ReSA task list > Foundation Data > Data Loading > Manage Data. The manage data screen opens. Through this screen, the user can choose the General Ledger template to maintain the data using spreadsheet download and upload.

When SYSTEM_OPTIONS.FINANCIAL_AP is A, the upload validates entries of valid segment combinations.



Configuring Drill Back and Forward Web Services

Retail web services table, RETAIL_SERVICE_REPORT_URL, must be updated with appropriate URLs to integrate with PeopleSoft Enterprise Financials.

- The records in the table for Services (indicated by RS_TYPE=S) for Account Validation (RAV) and Drill Forward (RDF), must be updated with the URL information from AIA where the services are hosted.

  Note: If Web services are secure, then the SYS_ACCOUNT column must be populated with authentication information in the form of *user name/password*.

- The records in the table for Reports (indicated by RS_TYPE=R) for both RMS and ReIM reports , must be updated with the URL information from the BIP Server where the reports are hosted.

ReIM Data Setup and Configuration

This section describes setup considerations for ReIM data.

System Options

As part of the RMS system options setup script, set the following options as indicated:

- FINANCIAL_AP =A

CURRENCY PRECISION

Because PeopleSoft Enterprise Financials uses only three decimals, the transactions generated by the Oracle Retail ReIM application must not include more than three decimals.

Make sure that the RMS table CURRENCIES has maximum of 3 decimal places for the currency_cost_dec attribute for difference currency codes.

The currency_rates table in RMS should be loaded initially by PeopleSoft Enterprise Financials.

Chart of Accounts Setup

The chart of accounts is set up manually in Oracle Retail applications and in PeopleSoft General Ledger. All account combinations are set up in each Set Of Books. Account validation is done while executing Financial Posting batch.

Segment Mapping

The retailer determines how many segments are populated. Up to 20 account segments can be specified. The following is an example of how segments are mapped between the ReIM transaction table and PeopleSoft General Ledger:

Table 11–3

ReIM Segments	PeopleSoft General Ledger Chart of Accounts
Segment 1	ACCOUNT
Segment 2	ALTACCT
Segment 3	DEPTID
Segment 4	OPERATING_UNIT
Segment 5	PRODUCT
Segment 6	FUND_CODE
Segment 7	CLASS_FLD
Segment 8	PROGRAM_CODE
Segment 9	BUDGET_REF
Segment 10	AFFILIATE
Segment 11	AFFILIATE_INTRA1
Segment 12	AFFILIATE_INTRA2
Segment 13	CHARTFIELD1
Segment 14	CHARTFIELD2
Segment 15	CHARTFIELD3
Segment 16	RESOURCE_TYPE
Segment 17	RESOURCE_CATEGORY
Segment 18	RESOURCE_SUB_CAT
Segment 19	BUSINESS_UNIT_PC
Segment 20	PROJECT_ID

If any one of the values in the 20 segments does not match the PeopleSoft General Ledger, the account combination is considered as invalid. The following error message is added to IM_POSTING_DOC_ERRORS table: "One or more accounts subjected to posting is invalid."

Segments 1 and 2 may be set up as dynamic at the Location level, or Segments 4 and 5 can be dynamic at the Department and Class level respectively. Segments defined as dynamic are allowed to be null for certain types of Basic Transaction or Reason Code cross-reference types. When a segment is null, the segment is assigned dynamically when transactions are posted. (Non-dynamic segments cannot be blank). Validation applies to the segment combination, not to individual segments.

Note: For Tran code TAP, each segment must have a value regardless of whether the segment is dynamic.

Running the Initial Load from PeopleSoft Financials

The initial load for ReIM is run by PeopleSoft and includes the following information:

- Suppliers
- Currency Rates
- Payment Terms

Note: The view, mv_currency_conversion_rates should be refreshed once the initial loads of currencies from PeopleSoft General Ledger are loaded to ReIM.

IM_SYSTEM_OPTIONS Table Setup

To accommodate integration, IM_SYSTEM_OPTOINS table should be configured with the following properties.

Note: Drill forward functionality is applicable for PeopleSoft - RMS and ReIM integration. Drill backward functionality is only applicable for PeopleSoft - RMS integration.

WS_FIN_ACC_VALID_URL - This attribute holds the URL for the financial account validation web service.

WS_FIN_ACC_VALID_CRED - This attribute indicates if the account validation web service call is authenticated or not. Y means Authentication enabled. N means Authentication is not enabled (See configuring web service credentials in weblogic server).

WS_FIN_DRILL_FWD_URL - This attribute holds the URL for the financial drill forward web service.

WS_FIN_DRILL_FWD_CRED - This attribute indicates if the drill forward web service call is authenticated or not. Y means Authentication enabled. N means Authentication is not enabled (See configuring web service credentials in weblogic server).

The above attributes are configured in the System Options screen.

Reports are created by Business Intelligence Publisher for the following:

The URL for each report must be updated in the table, retail_service_report_url. The following table provides sample URLs.

Reporting

Reports are created by Business Intelligence Publisher for the following:

- Merchandise Invoice
- Non-Merchandise Invoice
- Credit Note
- Credit Memo

- Debit Memo
- Receipt Write-Off

The URL for each report must be updated in the table, retail_service_report_url. The following table provides sample URLs:

Table 11–4

Document Type	Report Name	Sample Report URL
MRCHI	Merchandise invoice document Report	http://hostname:portno/xmlpserver_nonsso/ Guest/REIM13/Finance/invreport/invreport.xdo
NMRCHI	Non-Merchandise invoice document Report	http://hostname:portno /xmlpserver_nonsso/ Guest/REIM13/Finance/invreport/invreport.xdo
CRDNT	Credit Note document Report	http://hostname:portno/xmlpserver_nonsso/ Guest/REIM13/Finance/invreport/crnreport.xdo
CRDMEC	Credit Memo cost document Report	http://hostname:portno/xmlpserver_nonsso/ Guest/REIM13/Finance/invreport/memoreport.xdo
CRDMEQ	Credit Memo quantity document Report	http://hostname:portno/xmlpserver_nonsso/ Guest/REIM13/Finance/invreport/memoreport.xdo
DEBMEC	Debit Memo cost document Report	http://hostname:portno/xmlpserver_nonsso/ Guest/REIM13/Finance/invreport/imemoreport.xdo
DEBMEQ	Debit Memo quantity document Report	http://hostname:portno/xmlpserver_nonsso/ Guest/REIM13/Finance/invreport/memoreport.xdo
DEBMEV	Debit Memo Tax document Report	http://hostname:portno/xmlpserver_nonsso/ Guest/REIM13/Finance/invreport/memoreport.xdo
RWO	Receipt Write Off document Report	http://hostname:portno/xmlpserver_nonsso/ Guest/REIM13/Finance/invreport/rworeport.xdo

ReIM Transactional Maintenance

Integration to PeopleSoft General Ledger includes a number of transactions, as described below.

Calculation of TRANS_AMOUNT

The TRANS_AMOUNT field in the im_financial_stage table stores the value of the journal entry to be posted to PeopleSoft General Ledger. (The currency for the calculated amount is the currency assigned to the transaction). The TRANS_AMOUNT value is calculated as follows:

Table 11–5

Row Description	DEBIT_CREDIT_IND	TRANS_AMOUNT Value
Normal	Debit	Transaction Amount
Normal	Credit	(-1) * Transaction Amount

Table 11–5 (Cont.)

Row Description	DEBIT_CREDIT_IND	TRANS_AMOUNT Value
VAT	Debit	Transaction Amount * Tax Rate
VAT	Credit	(-1) * Transaction Amount * Tax Rate

Note: Transaction Amount is taken from the database column, IM_FINANCIALS_STAGE.AMOUNT.

Generation of Outgoing Data

A staging table accommodates the outgoing transfer of data. The reference key assigned to each document or receipt is used to find data on this table.

Outgoing Data

Table 11–6

From	To	Transactions
ReIM	PeopleSoft Accounts Payable	Invoices Debit Memos Credit Memos Credit Notes
ReIM	PeopleSoft General Ledger	General Ledger accounting entries resulting from the Invoice Matching process, including: Pre-paid invoices Receipt Write-offs
RMS	PeopleSoft General Ledger	Accounting entry data (potentially very high volume)
ReSA	PeopleSoft General Ledger	Accounting entry data (potentially very high volume)

Validation of Accounts When Posting Financial Entries

Valid accounts are stored in the ReIM table, IM_VALID_ACCOUNTS, which includes the Set of Books ID (sob_id) and 20 segments. An ORFI Web service validates accounts against the PeopleSoft Enterprise Financials system. Valid accounts are posted to IM_VALID_ACCOUNTS; invalid accounts are posted to IM_POSTING_DOC_ERROR. The following steps describe the validation process:

1. The ReIM system invokes the account validation Web service to validate the account. (A URL for the ORFI Web service is configured in the Reim SYSTEM_OPTIONS table.)
2. The posting batch job checks the accounts to be posted against the IM_VALID_ACCOUNTS table.
3. If the account entries are in the table, the transaction is posted to the G/L or AP tables.

4. If the account does not exist in the table, a collection of accounts is built. These collected accounts are validated against the PeopleSoft Enterprise Financials system, and a status is returned.
   - If the status of the collected accounts is valid, the accounts are inserted in the IM_VALID_ACCOUNTS table, and the transactions are posted to the staging tables.
   - If the status of the accounts is NOT valid, the entire collection is flagged as errors, and transactions are posted to IM_POSTING_DOC_ERROR.

Note: ReIM completes the first level of account validation and posts the transaction to staging tables. It is assumed the second level of account validation is done at the end of the extraction process (where transactions are moved from ReIM staging tables to PeopleSoft). If account validation fails at this point, Oracle Data Integrator (ODI) or PeopleSoft must change the account information before transactions are loaded to PeopleSoft, and the account change must be communicated to ReIM.

Maintenance of Valid Accounts

As account information is changed in the PeopleSoft system, the same changes are communicated to, and manually completed in, the ReIM system. After ReIM is updated accordingly, the AccountWorkspacePurge Batch is run to clear the valid accounts maintained locally in ReIM.

The AccountWorkspacePurge Batch can clear all valid accounts in the IM_VALID_ACCOUNTS table or only those that are considered updates in PeopleSoft.

Usage:

```
AccountPurge batch-alias-name  PURGE [ALL | <Accounts>]
```

Where

The first argument is batch alias name.

The second argument is the word PURGE.

The third argument is either ALL or specific accounts to be deleted from the local table.

Building and Posting Reference IDs

Drill back and drill forward functionality uses Reference ID to locate documents and receipts. A Reference ID is a combination of document type and document (or receipt) ID, as illustrated in the table below:

Table 11–7

Type	Doc ID	Receipt ID	Reference ID
Merchandise Invoice	101	Null	MRCHI#101
Non-Merchandise Invoice	102	Null	NMRCHI#102
Receipt	Null	103	RECEIPT#103

For documents, the Resolution Posting Batch program builds the Reference ID using the standard, Document Type + DeLimiter + Doc_id. For receipts, the program builds the Reference ID using the standard, Document Type + DeLimiter + Receipt_id.

To enable drill down functionality, Reference IDs are loaded to staging tables. FinancialsAPStageDao and FinancialsGLStageDao are populated, as are IM_RWO_SHIPMENT_HIST and IM_RWO_SHIPSKU_HIST.

Drilling Back

Drilling back allows users to view the source of posted PeopleSoft transactions that originated in Oracle Retail systems (from a voucher to an invoice, for example).

When drilling back from PeopleSoft Enterprise Financials, users are not directed to an actual screen within RMS, ReIM or ReSA. Rather, a retail Web service generates and launches a URL to a BI Publisher report. The report contains the information that typically appears on the appropriate retail screen.

Depending on the information requested by the user, PeopleSoft invokes web service in the ORFI layer which in turn calls the report locator service available in RMS. This service returns back BIP report reference URL which is passed back to PeopleSoft and report gets launched in browser window.

Information from the reference key determines what kind of report URL to issue. For example, if the retail key has a prefix of ReIM, the ReIM_REPORT_URL function is called, else the RMS_REPORT_URL function is called to retrieve the appropriate report URL. If the key does not match any key in the retail systems, an error message is launched.

Drilling Back to RMS and ReSA from PeopleSoft Enterprise Financials

The following function determines which RMS report to return to the user:

```
RMS_REPORT_URL() -
     O_error_message     IN OUT     RTK_ERRORS.RTK_TEXT%TYPE
     O_rpt_url           IN OUT     RETAIL_SERVICE_REPORT_URL.URL%TYPE
     I_ref_key           IN         KEY_MAP_GL.REFERENCE_TRACE_ID%TYPE
```

The appropriate report URL is found and issued as follows:

1. The ref_trace_type is found on KEY_MAP_GL by matching I_ref_key with the KEY_MAP_GL.REFERENCE_TRACE_ID column.
2. When ref type is determined, the re_trace_type is used to find the appropriate report URL on the RETAIL_SERVICE_REPORT_URL table.
3. The value of I_ref_key is appended to the end of the URL retrieved from the table.
4. The URL is sent back to the calling function.
5. If I_ref_key does not exist on KEY_MAP_GL, an error message is sent back to the calling function.

Drilling Back to ReIM from PeopleSoft Enterprise Financials

The following drill back options are available for viewing information within the ReIM system:

- Using Document ID, users can drill back to ReIM to view information related to a voucher or payment. The report includes information from im_doc_head and im_invoice_detail, the same data shown on the Document Maintenance Header screen within ReIM.
- Using the Receipt ID, users can drill back to view information from the Receipt Write-off History screen. Receipt write-offs occur either when an open receipt is

closed in ReIM or if a receipt is purged in RMS before it is fully matched. Details come from the IM_RWO_SHIPMENT_HIST and IM_RWO_SHIPSKU_HIST tables.

The function below determines which of the two ReIM reports to return to the user:

```
REIM_REPORT_URL() -
     O_error_message     IN OUT     RTK_ERRORS.RTK_TEXT%TYPE
     O_rpt_url           IN OUT     RETAIL_SERVICE_REPORT_URL.URL%TYPE
     I_ref_key           IN         KEY_MAP_GL.REFERENCE_TRACE_ID%TYPE
```

The I_ref_key contains the reference ID, which ultimately determines the type of report required. The appropriate BI Publisher report URL is found on the RETAIL_SERVICE_REPORT_URL table.

In general, if the reference ID has a prefix of RECEIPT, the report type (RS_CODE) is RCPT. Otherwise, the report type is DOC. For example:

Table 11–8

Reference ID	Report Type (RS_CODE)
MRCHI#101	DOC
NMRCHI#102	DOC
RECEIPT#103	RCPT

The following is an example of a BI Publisher URL that is generated upon drilling back to PeopleSoft Enterprise Financials for information on an invoice in ReIM, using Document ID as the search parameter:

```
http://myserver:myport/BIPublisher/Guest/ReIM/13.0.3/doc/tsf_det.xdo ?doc_id=101
```

Where

- http://myserver:myport/BIPublisher = the BI Publisher application server address and port
- Guest/ReIM/13.0.3 = the directory/folder location
- doc/tsf_det.xdo ? = report name (Document Report)
- doc_id=101 = the parameter name and value (Document ID 101)

The following is an example of an Oracle Business Intelligence Publisher URL that is generated upon drilling back to PeopleSoft Enterprise Financials for information on an invoice in ReIM, using Receipt ID as the search parameter:

```
http://myserver:myport/BIPublisher/Guest/ReIM/13.0.3/doc/tsf_det.xdo ?receipt_
id=101
```

Where

- http://myserver:myport/BIPublisher = the BI Publisher application server address and port
- Guest/ReIM/13.0.3 = the directory/folder location
- doc/tsf_det.xdo ? = report name (Receipt Report)
- receipt_id=101 = the parameter name and value (Receipt ID 101)

Drilling Forward

Drilling forward allows users to see detailed information about retail transactions that have been posted to PeopleSoft Enterprise Financials. When drilling forward, users are directed to selected "view-only" screens.

Drilling Forward From RMS/ReSA to PeopleSoft Enterprise Financials

The following forms may be used to drill forward from RMS/ReSA:

- RMS StartMenu->Finance->Transaction Data View (trandata.fmb)
- RMS StartMenu->Ordering->Fixed Deals->Fixed Deal Transaction Data View (fdltrandata.fmb)
- RMS StartMenu->Action->Sales Audit->Sales Audit Transaction Data View (satrandata.fmb)

Drilling Forward From ReIM to PeopleSoft Enterprise Financials

For drilling forward, the ORFI Web service uses the Invoice ID and Accounting Entry parameters. The ReIM system uses these parameters together as the Reference ID.

- From the Document Search screen, users can drill forward to PeopleSoft Enterprise Financials accounts payable to view voucher and payment status. The information is displayed on a read-only Payment Doc Status Inquiry screen. Drill forward access to the accounts payable system is available only for pre-paid invoices (but not for manually pre-paid invoices).

  To drill down to the payables/ledger screens, the user invokes the Web service as follows:

  ```
  Invoice ID/Accounting Entry parameter = Reference ID
  ```

- Drill forward access to the G/L system is available only for pre-paid and manually pre-paid invoices.

Note: For more information on drilling forward, see the *Oracle Retail Financial Integration for Oracle Retail Merchandise Operations Management and Oracle Financials Suite (E-Business Suite Financials or PeopleSoft Financials)*.

12

System Configuration

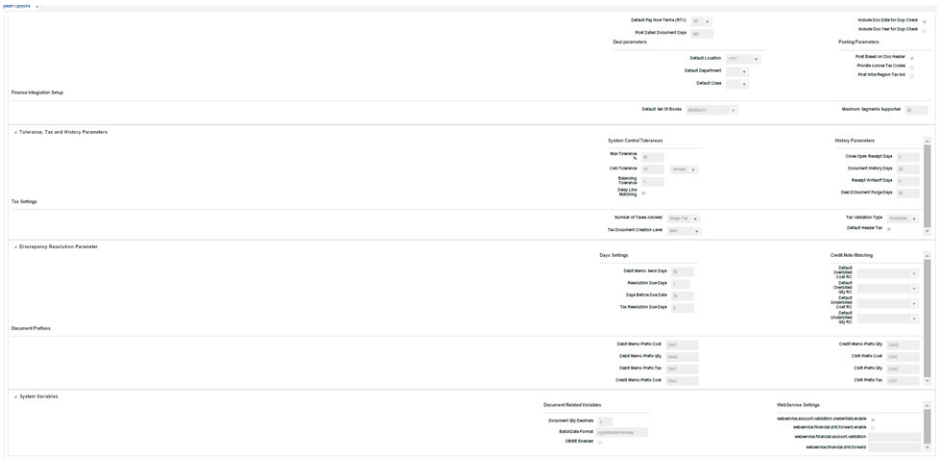
After installation, some of the mandatory steps need to be performed for the application to be functioning correctly. The objective of the checklist is to make sure that all the required data elements have been provided to eliminate failure in the document processing lifecycle.

Define System Options

Out of the box some of the system options have been set to default values. You need to verify that the values meet your business requirements. Some of the attributes that would need to reference identifiers for the existing entities could not be defaulted and would need to be set explicitly. Such attributes are:

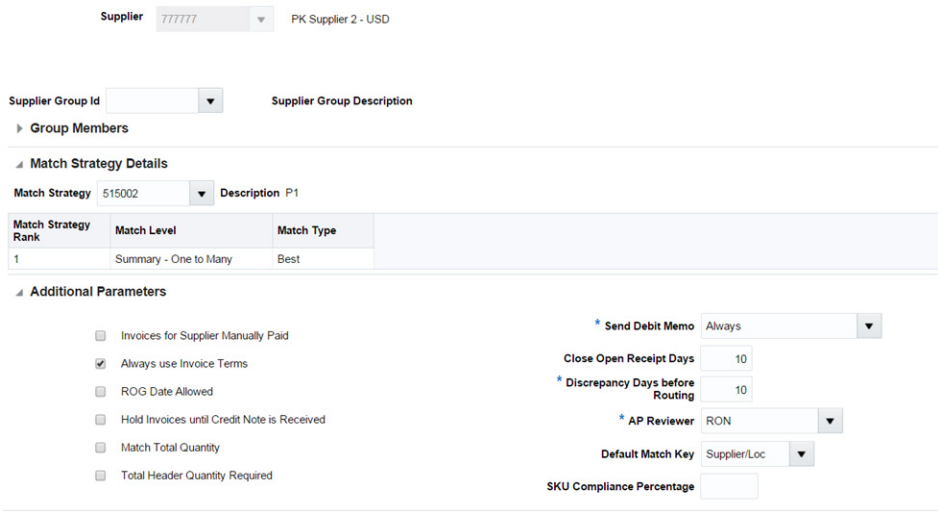
- Default Pay Now Terms (Deals)
- Default Pay Now Terms (RTV)
- Default Location
- Default Department
- Default Class
- Default Set Of Books
- URL for account validation web service
- URL for drill forward web service

Please see the *Oracle Retail Invoice Matching User Guide*, System Options section for more information.



Define Supplier Options

Each supplier in Retail Merchandising System would need to have Supplier Options defined before it can be "visible" within the Invoice Matching application. Supplier options can be defined at the Supplier level, Supplier Site level and Supplier Group level. Only Supplier level options are mandatory.



Please see the *Oracle Retail Invoice Matching User Guide*, Supplier Options section for more information.

Define Matching Tolerance

If you expect to match documents within tolerance rather than matching exactly, you would need to define tolerance entities and map tolerances to appropriate tolerance nodes. Please see the User Guide, Tolerance Maintenance and Tolerance Mapping sections for more information.

Define matching Strategies

If you expect to run Auto Match batch process, and you probably would, you would need to define some Matching Strategies. Please see the User Guide, Match Strategy Maintenance section for more information.

Define Reason Codes

Reason codes would need to be defined to perform discrepancies resolutions. Reason codes are synonyms for the resolution actions that the application user can take to resolve matching discrepancies. This release of Invoice Matching application doesn't provide a dedicated screen for reason code maintenance. As such, the reason codes would need to be defined via back-end. The reason codes are defined in IM_ REASON_CODES table.

An example reason code script to populate the table would look similar to the following:

```
INSERT INTO IM_REASON_CODES ( REASON_CODE_TYPE
                            , REASON_CODE_ID
                            , REASON_CODE_DESC
                            , ACTION
                            , COMMENT_REQUIRED_IND
                            , HINT_COMMENT
                            , DELETE_IND
                            , CREATED_BY
                            , LAST_UPDATED_BY )
 SELECT 'CNT'
 AS     REASON_CODE_TYPE
      , NVL ( ( SELECT MAX ( TO_NUMBER ( REASON_CODE_ID ) ) + 5
                  FROM IM_REASON_CODES )
            , 100 ) AS REASON_CODE_ID
      , 'CN TAXES ARE INCORRECT' AS REASON_CODE_DESC
      , 'CNRTF'                  AS ACTION
      , 'N'                      AS COMMENT_REQUIRED_IND
      , NULL                     AS HINT_COMMENT
      , 'N'                      AS DELETE_IND
      , 'REIM_DEMO_USER'         AS CREATED_BY
      , 'REIM_DEMO_USER'         AS LAST_UPDATED_BY
   FROM DUAL;
```

You can also consult *im_demo_data.sql* script that is shipped with the product. Please note that the script would need to be modified to suite your needs.

There should be at least one entry for each action that would need to be performed. Actions are subdivided based on categories (REASON_CODE_TYPE).

- Cost ('C')
- Quantity ('Q')
- Credit Note Matching related ('CNT')
- Return to vendor ('RTV')
- Tax ('T')

The same action can be mapped to more than one reason code. If you would need some comments to be provided when using the reason code, set the COMMENT_REQUIRED_IND to 'Y'. Hint comment can be optionally provided.

In addition to populating IM_REASON_CODES table with the reason codes, you would also need to populate IM_SEC_GRP_REASON_CODE table for each reason code. This table assigns reason codes to the security user groups defined in RMS. Only the reason codes that have been assigned to the user's security group will be available for processing. There should be a record for each user group/reason code that would require reason code access.

An example reason code security script to populate the table would look like:

```
INSERT
INTO
  IM_SEC_GRP_REASON_CODE
SELECT
  SG.GROUP_ID,
  IRC.reason_code_id
FROM
  im_reason_codes IRC,
  SEC_GROUP SG
WHERE
  IRC.delete_ind  ='N'
AND SG.GROUP_NAME = 'REIM_GROUP';
```

The example is also available in the *im_demo_data.sql* script.

See the *Oracle Retail Invoice Matching Data Model* for more information on the tables structure.

Define GL Mappings

To be able to successfully post transactions to Financial System, such as EBS you would need to define General Ledger account mappings. This release of Invoice Matching application doesn't provide a dedicated screen for General Ledger mappings. As such, the mappings would need to be defined via back-end. The mappings are defined in IM_GL_CROSS_REF table.

An example GL Cross Ref script to populate the table would look similar to the following:

```
INSERT
INTO
  IM_GL_CROSS_REF
  (
    ACCOUNT_TYPE,
    ACCOUNT_CODE,
    SEGMENT_NO,
    SEGMENT_VALUE,
    SET_OF_BOOKS_ID,
    TAX_CODE,
        CREATED_BY,
        LAST_UPDATED_BY
  )
SELECT
  'BT'                AS ACCOUNT_TYPE,
  'TAX'               AS ACCOUNT_CODE,
  10                  AS SEGMENT_NO,
  'BTTAXSEG10'        AS SEGMENT_VALUE,
  FGS.SET_OF_BOOKS_ID AS SET_OF_BOOKS_ID,
  'NOCODE'            AS TAX_CODE,
  'REIM_DEMO_USER'    AS CREATED_BY,
  'REIM_DEMO_USER'    AS LAST_UPDATED_BY
FROM
  FIF_GL_SETUP FGS;
```

The example also is available in im_demo_data.sql script.

The application supports up to 20 segments. Not all segments are required. Usually the number of segments doesn't exceed 10. There should be a separate set of mappings for each set of books defined in the system. Mappings would need to be defined for all basic transactions, as well as for all non-merchandising codes and for all reason codes defined in the previous step.

Optionally, the mappings can be differentiated per tax code.

In addition to populating IM_GL_CROSS_REF table with the GL mappings, you would also need to populate IM_GL_OPTIONS to indicate what segments, if any, would be dynamic. Dynamic segments don't need explicit mappings, but rather the mappings are dynamically determined based of the segment meaning.

An example GL Options script to populate the table will look similar to the following:

```
INSERT
INTO
  IM_GL_OPTIONS
  (
    SEGMENT_NO,
    DYNAMIC_IND,
    SET_OF_BOOKS_ID,
    SEGMENT_LABEL,
    BUSINESS_ATTRIBUTE,
    CREATED_BY,
    LAST_UPDATED_BY
  )
SELECT
  '10'                AS SEGMENT_NO,
  'N'                 AS DYNAMIC_IND,
  FGS.SET_OF_BOOKS_ID AS SET_OF_BOOKS_ID,
  'Future4'           AS SEGMENT_LABEL,
  'Future4'           AS BUSINESS_ATTRIBUTE,
  'REIM_DEMO_USER'    AS CREATED_BY,
  'REIM_DEMO_USER'    AS LAST_UPDATED_BY
FROM
  FIF_GL_SETUP FGS;
```

The example also is available in im_demo_data.sql script.

If Location segments are defined as dynamic then location segment mapping would need to be provided. It is done by populating IM_DYNAMIC_SEGMENT_LOC table. There should be an entry for each location that would need to be posted. There should be a separate set of mappings for each set of books defined in the system.

An example location mapping script to populate the table will look similar to the following:

```
INSERT
INTO
  IM_DYNAMIC_SEGMENT_LOC
  (
    LOCATION,
    LOC_SEGMENT,
    COMPANY_SEGMENT,
    CREATED_BY,
    LAST_UPDATED_BY
  )
SELECT
  STORE                          AS LOCATION,
  ('DEMOS10000000' ||STORE)      AS LOC_SEGMENT,
  'DEMO12345678910'              AS COMPANY_SEGMENT,
  'REIM_DEMO_USER',
  'REIM_DEMO_USER'
FROM
  STORE
UNION
SELECT
  WH                             AS LOCATION,
  ('DEMOW20000000'|| WH)         AS LOC_SEGMENT,
  'DEMO12345678910'              AS COMPANY_SEGMENT,
  'REIM_DEMO_USER',
  'REIM_DEMO_USER'
FROM
  WH
WHERE
  WH = PHYSICAL_WH;
```

If Department or Class segments are defined as dynamic then merchandising hierarchy segment mapping would need to be provided. It is done by populating IM_DYNAMIC_SEGMENT_DEPT_CLASS table. There should be an entry for every department/class combination that would need to be posted. There should be a separate set of mappings for each set of books defined in the system.

An example department/class mapping script to populate the table will look similar to the following:

```
INSERT
INTO
  IM_DYNAMIC_SEGMENT_DEPT_CLASS
  (
    DEPT,
    CLASS,
    DEPT_SEGMENT,
    CLASS_SEGMENT,
    SET_OF_BOOKS_ID,
    CREATED_BY,
    LAST_UPDATED_BY
  )
SELECT
  C.DEPT                                AS DEPT,
  C.CLASS                               AS CLASS,
  ('DEMOD33333333' || C.DEPT)           AS DEPT_SEGMENT,
  ('DEMOC44444444' || C.CLASS)          AS CLASS_SEGMENT,
  FGS.SET_OF_BOOKS_ID                   AS SET_OF_BOOKS_ID,
  'REIM_DEMO_USER',
  'REIM_DEMO_USER'
FROM
  CLASS C, FIF_GL_SETUP FGS;
```

See the *Oracle Retail Invoice Matching Data Model* for more information on the tables structure.

13

Batch Processes

This chapter provides the following:

- An overview of the batch architecture
- A functional summary of each batch process, along with its dependencies
- A description of some of the features of the batch processes (batch return values, batch threading, and so on)
- Development designs for each batch process

Batch Architectural Overview

ReIM batch processes are run as Java applications. Batch processes engage in a shared processing with the UI in a client server model.

Services retrieve the data on which the batch processes work to complete their tasks. As noted in "Chapter 2, "Technical Architecture" the service layer consists of a collection of Java classes that implements business logic (data retrieval, updates, deletions, and so on) through one or more high-level methods.

The business logic occurs within the service code, while the technical processing occurs within the batch code.

Note the following characteristics of the ReIM batch processes:

- They are not accessible through a graphical user interface (GUI).
- They are scheduled by the retailer.
- They are designed to process large volumes of data. However, the volume can be managed using the inclusion-exclusion configuration feature (see section Batch Configuration)
- ReIM batches should run only in the batch window when no users exist in the system. This requirement is related to locking consideration (see locking section for more information).
- They run in a client server model. Client side code is a plain java which basically responsible for the validation of input parameters, retrieval of user credentials for the given alias name.
- In case there is no valid user/password pair is found for the given alias name, the client program complains the same and terminates. On the successful identification of credentials, client program makes a call to execute<<Batch>> method of corresponding EJB on the application server

Batch Process Configuration

For programs in the financial posting batch cycle, we have implemented a new configuration feature. The volume of data processed by each batch execution can be controlled through a table configuration. While executing a particular batch program (say AUTOMATCH), we may restrict the batch candidates to either a particular supplier or exclude a particular supplier to manage the batch load. At any point of time, a max of one exclude or one include should be configured per a particular day.

A new table IM_BATCH_CONFIG is included in the data model which is used for configuring the exclusion or inclusion of suppliers per batch for a particular day. An indicator called 'PROCESSED_IND' is in the table that tells if the inclusion/ exclusion are already processed or not. Its initial value is N.

Table 13–1 IM_BATCH_CONFIG Parameters

Parameter	Description
BATCH_NAME	Used for configuring the batch name.
SUPPLIER	The supplier that the batch should exclude/include.
SCOPE	Exclude (E)/Include (I)
PROCESSED	Initial value N. Once the batch completes, program sets it to Y so that it would not considered next execution.
PROCESSED_DATE	This attribute is added for the auditing purpose. It also cater as a constraint to make sure that only one Exclude or Include are configured per supplier per day.

The query that picks the candidates for the each batch is modified to join with IM_ BATCH_CONFIG table to first exclude the documents/data that are of SUPPLIERS configured to exclude in the particular batch program with a PROCESSED value N. Then the inclusion logic is applied, if any. If no exclusions/inclusions have been configured for a particular batch with PROCESSED value N, then the batch will process all the eligible data.

The batches that have this capability are

- Automatch batch
- Financial Posting batch
- Resolution Action Rollup batch
- Complex Deal Upload
- Fixed Deal Upload

EDI-Related File-Based Batch Processes

ReIM EDI-related batch processes are file based. For example, they either input a flat file into the system (EDI Injector) from outside the system, or they output a flat file from the system (EDI invoice download) to be sent to another system (that of a vendor). Both the EDI Injector and the EDI invoice download batch processes are described later in this chapter. For the EDI Injector batch, the input file/folder and the rejection file/folder should be on the same physical machine as the application server is running on. This is a requirement from the file system access permission stand point. Similarly, the EDI download batch process can save the edi output file only to a folder that is on the same machine as the server is running on.

Internal Batch Processes

Other batch processes within ReIM do not input or output files. Rather, the goal of these batch processes is to take a snapshot of potentially large amounts of data from the key tables within the database, transform that data through processing, and then return it.

Internal batch processes that are described later in this chapter include:

- Auto-match
- Batch purge
- Account purge
- Reason Code action rollup

Internal Batch Processes that Write to Staging Tables

The third type of batch process within ReIM takes a snapshot of potentially large amounts of data from the key tables within the database, transforms that data through processing, and then writes that data to staging tables.

This communication process has been designed with the assumption that, during production, ReIM will reside within the same database as the merchandising system. Presumably, during implementation, the retailer will develop an optimum way to move the applicable data from the staging tables to the appropriate location for that data.

The internal batch processes that write to staging tables are described later in this chapter.

Batch Processes that Extract from Merchandising System (RMS) Staging Tables

The fourth type of batch process within ReIM extracts data from merchandising system staging tables, create documents with the data, and write the data to ReIM tables. The batch processes that follow this processing pattern include the following:

- Complex deal upload
- Fixed deal upload

Batch Names

The following table describes ReIM batch processes. The table order reflects the dependencies that exist among the ReIM batch processes but does not include any dependencies that exist between ReIM and the merchandising system with which it interacts.

Table 13–2 Batch Names

Batch Name	Class (oracle.retail.apps.reim.batch.client)
Tables purge	TablesPurgeBatchClient
Account purge	AccountWorkspacePurgeBatchClient
EDI Injector	EdiInjectorBatchClient
Auto-match	AutoMatchBatchClient
Receipt write-off	ReceiptWriteOffBatchClient
Reason code action rollup	ReasonCodeActionRollupBatchClient

Table 13–2 (Cont.) Batch Names

Batch Name	Class (oracle.retail.apps.reim.batch.client)
Financial posting	FinancialPostingBatchClient
EDI Invoice download	EdiDownloadBatchClient
Complex deal upload	ComplexDealUploadBatchClient
Fixed deal upload	FixedDealUploadBatchClient
Financial Posting workspace Purge	FinancialPostingWorkspacePurgeBatchClient

Functional Descriptions and Dependencies

The following table summarizes ReIM batch processes and includes both a description of each batch process's business functionality and its batch dependencies:

Table 13–3 ReIM Batch Processes

Batch Processes	Details	Batch Dependencies
Batch purge	This process deletes data from database tables while maintaining database integrity. This process deletes records from the ReIM application that meet certain business criteria (for example, records that are marked for deletion by the application user, records that linger in the system beyond certain number of days, and so on).	
Account purge	This process deletes the accounts maintained locally in the ReIM application.	
EDI Injector	This batch process uploads merchandise, non-merchandise invoices, credit notes, debit memos, and credit note requests from the EDI into the invoice-matching tables.	
Auto-match	Auto-match is a system batch process that attempts to match invoices to receipts without manual intervention. Invoices that are in ready for match, unresolved, or multi-unresolved status are retrieved from the database to be run through the auto-match algorithm.	▪ EDI Injector ▪ Receipt upload (Merchandising system, such as RMS)
Receipt write-off	In order for retailers to track received goods not invoiced, they must have the ability to 'write-off' these goods for financial tracking. ReIM has a system parameter (which can be overwritten at the supplier level) defining the maximum amount of time an open, non-fully matched receipt will be available for matching. Every time the Receipt write-off process is run, each non-fully matched open receipt received date is compared with the current date minus the system parameter. If the received date is before this difference, the receipt is 'written-off,' and the invoice match status is closed.	Auto-match and any associated processing must run prior to this batch processing.

Table 13–3 (Cont.) RelM Batch Processes

Batch Processes	Details	Batch Dependencies
Reason code action rollup	This batch process sweeps the action staging table and creates debit memos, credit memos, and credit note requests as needed. Only a single debit or credit memo is created per invoice/discrepancy type, with line details from all related actions for the same discrepancy type. If hold invoice functionality is on, each generated document is assigned the invoice number to which it corresponds to ensure all related documents are released to accounts payable at the same time. This process deletes these records when completed; they are deleted after posting. Note that a separate, retailer-created batch process sweeps the receiver adjustment table. The action staging table is used during posting to post the reason code actions to the financial staging table. A separate, retailer-created batch process sweeps the receiver adjustment table. The process compares the unit cost and/or quantity received for the item on the shipment with the expected unit cost and/or quantity on the IM_RECEIVER_COST_ADJUST and/or IM_RECEIVER_UNIT_ADJUST tables. If a match exists, the receiver cost and/or unit adjustment has occurred in RMS (or the equivalent merchandising system). As a result, the process sets the 'pending adjustment' flag on IM_INVOICE_DETAIL table to false for the invoice line. The reason code actions are rolled up for an invoice only if no invoice lines on the invoice have any pending adjustments.	
Financial posting	A recurring resolution posting process retrieves all matched invoices and approved documents. If hold invoice functionality is used, then matched Credit Notes rather than approved Credit Notes are processed. Documents which should be sent to the Financial A/P system are sent to the AP staging tables: IM_AP_STAGE_HEAD, and IM_AP_STAGE_DETAIL. Transactions which should be sent to the Financial G/L system are sent to the IM_FINANCIALS_STAGE table	

Table 13–3 (Cont.) ReIM Batch Processes

Batch Processes	Details	Batch Dependencies
EDI invoice download	The EdiDownload module creates a flat file to match the EDI invoice download file format. The module retrieves all header, detail, and non-merchandise information and formats the data as needed. In other words, the EDI invoice download process retrieves debit memos, credit note requests, and credit memos in 'approved' status from the resolution posting process and creates a flat file. The client converts the flat file into an EDI format by the client and sends it through the EDI invoice download transaction set.	Auto-match must run prior to the EDI invoice download.
Complex deal upload	This module reads data from RMS staging tables, creates credit memos, debit memos, and credit note requests out of the data, and stores the supporting deal data on a ReIM table for later use during posting.	
Fixed deal upload	This module reads data from RMS staging tables, creates credit memos, debit memos, and credit note requests out of those, and stores the supporting deal data on a ReIM table for later use during posting.	

Features of the Batch Processes

This section describes the features of batch processes.

Scheduler and the Command Line

If the client uses a scheduler, batch process arguments are placed into the scheduler.

If the client does not use a scheduler, batch process parameters must be passed in at the UNIX command line.

Each of these scripts interacts with the generic shell script. These scripts take any and all arguments that their corresponding batch process would take when executing.

Batch Return Values

The following guidelines describe the batch process return values that ReIM batch processes utilize:

- SUCCESS = 0
- FAILED_INIT = 1
- FAILED_PROCESS = 2

Batch Log and Error File Paths

The client side log file location is determined by the retailer through the logj4.properties file. The errors that occur on the server side program will be written to the server log which can be configured by weblogic administrator. If an error occurs that causes a batch process to suddenly come to a complete halt, the system writes to the configured log file. See "Chapter 3, "Backend System Administration and Configuration" for more information.

Multi-Threading Batch Processes

The following batch processes shown below have multi-threading capabilities. The configuration related to some of the multi threaded batches can be configured in REIM System options. See "Chapter 3, "Backend System Administration and Configuration" for more information.

Complex Deal Upload (ComplexDealUploadBatch)

This process is threaded by a group (or bulk) of deals. Each group constitutes a thread.

Fixed Deal Upload (FixedDealUploadBatch)

This process is threaded by a group (or bulk) of deals. Each group constitutes a thread.

EDI Injector (EdiInjectorBatch)

This process is threaded into groups of documents. Each thread handles the business validation of the entire document group.

Auto-Match (AutoMatchBatch)

Auto-match is threaded based on the number of invoice-items and receipt-items involved in the match.

A Note about Restart and Recovery

Most ReIM batch processes do not utilize any type of restart and recovery procedures. Rather, if a restart is required, the process can simply be restarted, and it will start where it left off.

This solution is true for all batch processes other than those noted below:

- EDI injector (its restart and recovery methods is described in its design below).
- EDI invoice download (its restart and recovery methods is described in its design below).

Executing Batch Processes

Batch processes are executed through the batch client framework. This framework is responsible for ensuring that the batch job is passed the appropriate arguments. The arguments for the batch runner are as follows:

- Batch job class name
- batch-alias-name
- Batch arguments

> **Note:** Batches are run with an alias name rather than with a user name/password combination. The alias name is mapped to the user credentials inside a password store called a wallet.
>
> At run time the batches access the wallet and retrieve the user ID and password for authentication purposes.

Below is an example of how the batch runner would be utilized to execute the EdiInjectorBatch process:

```
reimediinjector batch-alias-name /dir/input.dat /dir2/output.dat
```

The batch client programs require the application libraries (JAR files) to be on the classpath in order to execute successfully.

Tables Purge Batch Design

The batch purging process deletes data from database tables while maintaining database integrity. This process deletes records from the ReIM application that meet certain business criteria (for example, records that are marked for deletion by the application user, records that linger in the system beyond certain number of days, and so on). The TablesPurge process does not generate any cascade relationships and/or SQL queries on the fly. The main features of the process are illustrated below:

Usage

The following arguments are applicable for the TablesPurgeBatch process:

reimpurge batch-alias-name [PURGE_OPERATIONALS | PURGE_WORKSPACES | PURGE_WORKSPACES_AND_OPERATIONALS]

The first argument is batch alias name. The second argument is the purge action. PURGE_OPERATIONALS would make the batch purge data from operational tables alone, PURGE_WORKSPACES would make the batch purge data from all workspace tables, PURGE_WORKSPACES_AND_OPERATIONALS would make the batch purge data from both operational and workspace tables. By default, the purge action is committed.

Purge Operational

Data from the operational tables will be purged based on the document history days system option. The tables are purged in the reverse order of their relationship to each other, history tables first and then the details and then the main or primary table.

Purge Workspace

The workspace tables used for display or internal calculation purposes for Search, document maintenance, Matching or Posting processes would be truncated. Indexes on all affected tables would be rebuilt after truncate. It is recommended that the workspace tables be truncated frequently, as they need to be well maintained to ensure optimal performance of the front end.

Purge Workspace and Operational

Data from the workspace tables are truncated followed by the deletion of data from the operational tables.

Primary Tables Involved

The following lists include the tables on which the purging algorithm is applied:

Operational

- IM_DOC_HEAD
- IM_INVOICE_DETAIL
- IM_RESOLUTION_ACTION

- IM_RECEIPT_ITEM_POSTING_%
- IM_%_MATCH_HISTORY

Workspace

- IM_%_SEARCH_WS
- IM_MATCH_%_WS
- IM_MATCH_POOL_%
- IM_POSTING_DOC_%

Other tables of less significance also get purged.

Accounts Purge Batch Design

This process deletes the accounts maintained locally in the ReIM application. The batch retrieves the accounts in IM_VALID_ACCOUNTS table and validates the account against the integrated financial system. Accounts that are invalid in the financial system are deleted from IM_VALID_ACCOUNTS table.

> **Note:** Run the batch whenever account information changes are communicated to ReIM.

Usage

The following arguments are applicable for the AccountWorkspacePurgeBatch process:

```
reimaccountworkspacepurge batch-alias-name
```

Major Modules

AccountWorkspacePurgeBatchClient

Major Tables

IM_VALID_ACCOUNTS

EDI Invoice Injector Batch Design

The EDI Injector Batch process performs the following:

- Reads each transaction within the file.
- Runs a file format validation (verifying file descriptors and line numbers; ensuring that numeric fields are all numeric and that character fields are all characters; looking for the invalid ordering of record type-THEAD followed directly by another THEAD; and so on). Certain file formatting errors cause the process to terminate with a message indicating the problem. A limited set of data validation errors can cause invalid EDI transaction to be fixable, where the data can be corrected through online process. The rest of the data validation errors cause the invalid transaction to be written to a set of reject files where a user must correct the problems and re-run the files.
- Validates the data against the ReIM system and the merchandising system (RMS).

- Any errors found are recorded in to the error table (IM_INJECT_DOC_ERROR) so that users can audit and fix any transactions that were rejected.
- Adds the data to the ReIM system. All valid transactions are written to the IM_ DOC_xxx, IM_INVOICE_xxxtables.
- The size of the Logical Unit of work for each chunk needs to be defined in ReIM's system options.

Usage

The following arguments are applicable for the EDI Injector Batch process:

```
reimediinjector batch-alias-name input-file/input-path output-file/output-path
```

Assumptions and Scheduling Notes

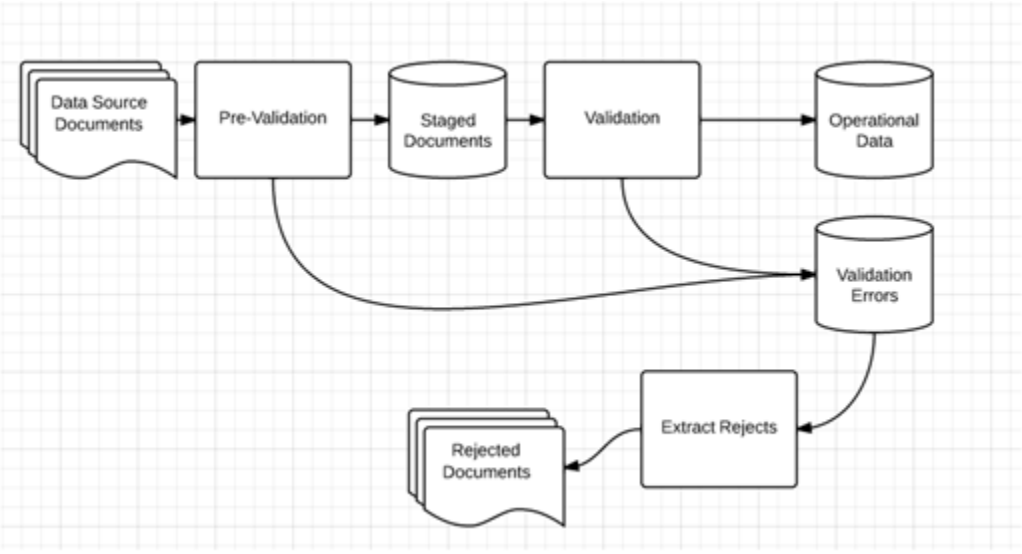
This process can be run ad-hoc but in general it should be run before the auto-match process.

Restart and Recovery

If the EDI Injector Batch aborts without processing an entire file, the file can simply be rerun. When this action is completed, there will be multiple errors for the transactions that were successfully uploaded and the other transactions will be uploaded at that time as well. If the cause of the aborted process is software related, this fix may not solve the issue. Other steps may be required to ensure that the process completes its entire initial run.

High-Level Flow Diagram

The following diagram offers a high-level view of the processing logic utilized within the EDI Injector Batch process.



Primary Tables Involved

The following tables are involved in the EDI Injector Batch process:

Operational Data Tables

- IM_DOC_HEAD
- IM_DOC_TAX
- IM_INVOICE_DETAIL
- IM_INVOICE_DETAIL_TAX
- IM_INVOICE_DETAIL _ALLOWANCE
- IM_INVOICE_DETAIL_ALLW_TAX
- IM_DOC_NON_MERCH
- IM_DOC_NON_MERCH_TAX
- IM_DOC_DETAIL_REASON_CODES
- IM_DOC_DETAIL_RC_TAX

Injector Workspace Tables

- IM_INJECT_DOC_DETAIL
- IM_INJECT_DOC_DETAIL_ALLOWANCE
- IM_INJECT_DOC_DETAIL_ALLOW_TAX
- IM_INJECT_DOC_DETAIL_TAX
- IM_INJECT_DOC_ERROR
- IM_INJECT_DOC_HEADER
- IM_INJECT_DOC_NON_MERCH
- IM_INJECT_DOC_NON_MERCH_TAX
- IM_INJECT_DOC_RECORD
- IM_INJECT_DOC_RULE
- IM_INJECT_DOC_TAX
- IM_INJECT_STATUS

Invoice Auto-Match Batch Design

Auto-match is a system batch process that attempts to match invoices to receipts without manual intervention. Invoices that are in ready-for-match, unresolved, or multi-unresolved status are retrieved from the database to be run through the auto-match algorithm.

The inputs into the auto-match process include the following:

- Invoices
- Receipts
- Purchase orders
- Match Strategy
- Tolerance

ReIM owns invoices, Match Strategy, and Tolerance while receipts and purchase orders are owned by a merchandising system, such as RMS.

The Match Strategy rules feature allows retailers to build and maintain match strategies which specifically define the types of matches which should be attempted

and the order in which they should be tried during the auto-match process. The match strategies can be defined at the system, supplier group or supplier level. The creation of a Supplier Group is tightly integrated to the logic for selecting documents to be processed by the match engine. If a Supplier Group is created, all the documents for all the suppliers in the group are considered by the match engine together. If a match strategy is defined at the Supplier Group level, then it is used to determine what match attempts to apply against the documents in the supplier group. If a match strategy is not defined at the supplier group level, then the system default match strategy is used to determine which match attempts are used to attempt to match documents for the supplier group. If a match strategy was set up for one of the suppliers for a supplier group, it is ignored by the match engine. If a supplier is not part of a supplier group, then all the documents for that supplier are considered by the match engine together. If a match strategy is defined at the Supplier level, then it is used to determine what match attempts to apply against the documents for that supplier. If a match strategy is not defined at the supplier level, then the system default match strategy is used to determine which match attempts should be used to attempt to match documents for the supplier.

The auto-match process attempts to match the invoices to receipts to the best of its abilities. The process assign different statuses according to the level of matching achieved.

If an invoice arrives prior to a receipt (for a particular PO), the auto-match process attempts only to match invoice unit cost to PO unit cost if Cost Pre-matching is opted while running the auto-match batch.

When a complete match cannot be made, manual intervention is required through online processes.

The size of the Logical Unit of work for each chunk needs to be defined in ReIM's system options.

Usage

The following arguments are applicable for the Invoice Auto-Match Batch process:

```
reimautomatch batch-alias-name
```

Algorithms

The following algorithms comprise the auto-match process:

Cost pre-matching

The Cost Pre-Matching routine is optional but if it is run, it runs as the first step of the Auto-match batch. When the Cost Pre-Matching routine is run, it is run against all suppliers. The routine is only executed if no receipt exists for the order on the invoice. If it finds differences in the cost on the order and the cost on the invoice which are outside of the tolerance level, it generates a cost discrepancy. If a match can be obtained, the invoice remains in ready-for-match status and is retrieved again for matching once the receipt comes in.

Summary matching

The Auto match batch attempts various types of Summary Matches based on the Match Strategy associated with the supplier or supplier group. Summary Matching involves looking at the total document values (cost and optionally quantity) without considering the specific items on each document.

Summary Match All-to-All

The all-to-all match attempts to match all invoice documents to all receipt documents in the match pool. Used in combination with the Match Strategy table, all-to-all matching provides the user:

- The option to choose whether or not to run the all-to-all match.
- The option to choose the order in which the all-to-all match is attempted. The user could decide to execute other match attempts before the all-to-all match.
- The option to decide how to group the invoices and receipts together to attempt matching by specifying the match key.

Summary Match One-to-Many

The one-to-many match attempts to match one invoice document to one or more receipt documents. There are two options when performing a one-to-many summary match:

- Regular Match

  Regular Match attempts to match the invoices and receipts in the pool as one to one matches. If an invoice could match to two or more receipts within tolerance, then the match fails. Similarly, if two or more receipts could match to a single receipt (within tolerance), then the match fails and both invoices are put in multi-unresolved status. If Regular Match fails because the invoice could be matched to multiple receipts or if it failed because multiple invoices could be matched to one receipt, the invoice is flagged as multi-unresolved. If a Regular Match fails for any other reason, the invoice is flagged as an 'Unresolved' match.

- Best Match

  The Best Match setting applies additional logic to select better matches when multiple receipts or receipt combinations can be matched to a single invoice. The best match process creates all combinations of one invoice to one or more receipts and selects the best match. The best match logic selects the receipt or combination of receipts that provides the lowest absolute variance. If two potential matches to the invoice have the same absolute variance but one is an overbill and one is an underbill, the underbill takes precedence. If the two potential matches have identical variances, then the invoice quantity matching will be used as an additional criteria. The match with the smallest absolute quantity variance, taken as the best match. If the absolute quantity variances are the same, then no best match can be determined, and the invoice is left as unmatched.

SKU Compliance on Summary Match

The SKU compliance feature can be used to match invoice items and values to what was actually received. SKU compliance is only calculated if all invoices in a match have details. Therefore, if any of the invoices in the match is a header only invoice, the SKU compliance is skipped. SKU compliance checks for how many of the items on the invoice(s) are on the receipt(s) and how many of the items on the receipt(s) are on the invoice(s). There is a percent calculation for each of these ratios, and both ratios must pass the SKU compliance percent for the match to be accepted.

Tax Validation on Header only Matches

ReIM uses a routine in the auto-matching program to perform a tax validation for header only invoices. The tax validation is executed when a header only invoice matches (either perfectly or within tolerance). The tax validation compares the taxes on the invoice to the taxes generated by the items from the receipt. In addition, the tax validation:

- ensures that all tax codes used on the invoice(s) are also used on the receipts in the match, and that the tax rates are exactly the same.
- ensures that all tax codes on the receipts used in the match are also on the invoice(s) and that the rates match.

If the match passes these two criteria, the invoice (and receipts) can be considered matched. If the validation fails, the invoice(s) are put into tax discrepant status.

Detail matching

In auto-matching, matching can be performed for entire invoices or broken down to the line level. Detail matching is performed by item.

- Eligibility for Detail Matching

  In order to be eligible for detail matching, an invoice or receipt must meet the following conditions:

  1. Item lines must be present on the invoice:
  2. The invoice or receipt must not be part of a manual group.

- Regular versus Best Match

  Regular detail matching compares the invoice item with the matching receipt item from all receipts in the pool. When regular matching is only done within a PO, the unit cost on all the receipt items is the same. If the match key being used allows the user to cross PO's, a constraint is included to require all receipt costs on an item to be the same. If receipt costs are different for the same item, detail matching is not allowed for the item.

  The Best Match Strategy for detail matching does two separate routines to attempt to match items within the match pool:

  1. If the cost of the item on all invoices in the pool is the same, Best Match attempts to match all invoices to all receipts in the pool for that item (an all-to-all match). If they are within tolerance for both cost and quantity in the all-to-all step, then the item is matched on all invoices and on all receipts within the pool. If either the cost or quantity match fails, then nothing is flagged as matched.
  2. For the item, look at the item on each invoice in the pool individually and compare it to the sum of all receipts in the pool and select the best match. The criteria for determining the best match in this scenario is as follows:
     - Calculate the unit cost variance, and if it is out of tolerance the invoice is rejected from best match consideration.
     - Calculate the quantity variance and if it is out of tolerance the invoice is rejected from best match consideration.
     - Calculate the variance on the extended cost between the invoice item and the receipt item(s). This variance is compared against all matches which pass the previous steps. Compare the absolute variance for all the matches which are eligible for best match consideration. Take the match with the least absolute variance as the best match. If two matches have the same absolute variance but one is an overbill and one is an underbill, select the underbill as the best match. If the variances are identical, then a best match is not possible, so the match is skipped.

  If the Best Match attempt is unsuccessful, it means that the Regular Match would also have not been successful. However, if the routing date has passed you should

attempt regular matching including the auto-resolution process and the generation of discrepancies.

The Regular Match attempts to match the invoice item with the receipt items from all receipts in the match key (receipt unit cost must all match the unit cost of the item on the PO(/loc) for the invoice being matched).

Generating Discrepancies

During regular detail matching, the auto-matching process generates discrepancies for cost and quantity discrepancies which are outside of tolerance.

The Tolerance table includes an Auto Resolution column which is used to determine the variance percent (or amount) allowed to complete an automatic resolution. The Auto Resolution column means that there are three types of discrepancies:

- Discrepancies which are within the 'variance within tolerance' (VWT) setting.
- Discrepancies which have a variance which can be automatically resolved.
- Discrepancies which have a variance which is too large to be resolved automatically. These variances generate a discrepancy and are sent to the Discrepancy Review list.

Automated Discrepancy Resolution

When a discrepancy has been identified as one which can be automatically resolved (based on comparing the variance with the applicable tolerance from the tolerance table), the system looks up the appropriate resolution action on the reason code table using the code assigned on the tolerance table row associated with the discrepancy. The resolution action is applied to resolve the discrepancy. If this is the last item on the invoice to be resolved then the whole invoice is flagged as matched. If this was the last item for the receipt to be resolved, then the receipt is also flagged as matched.

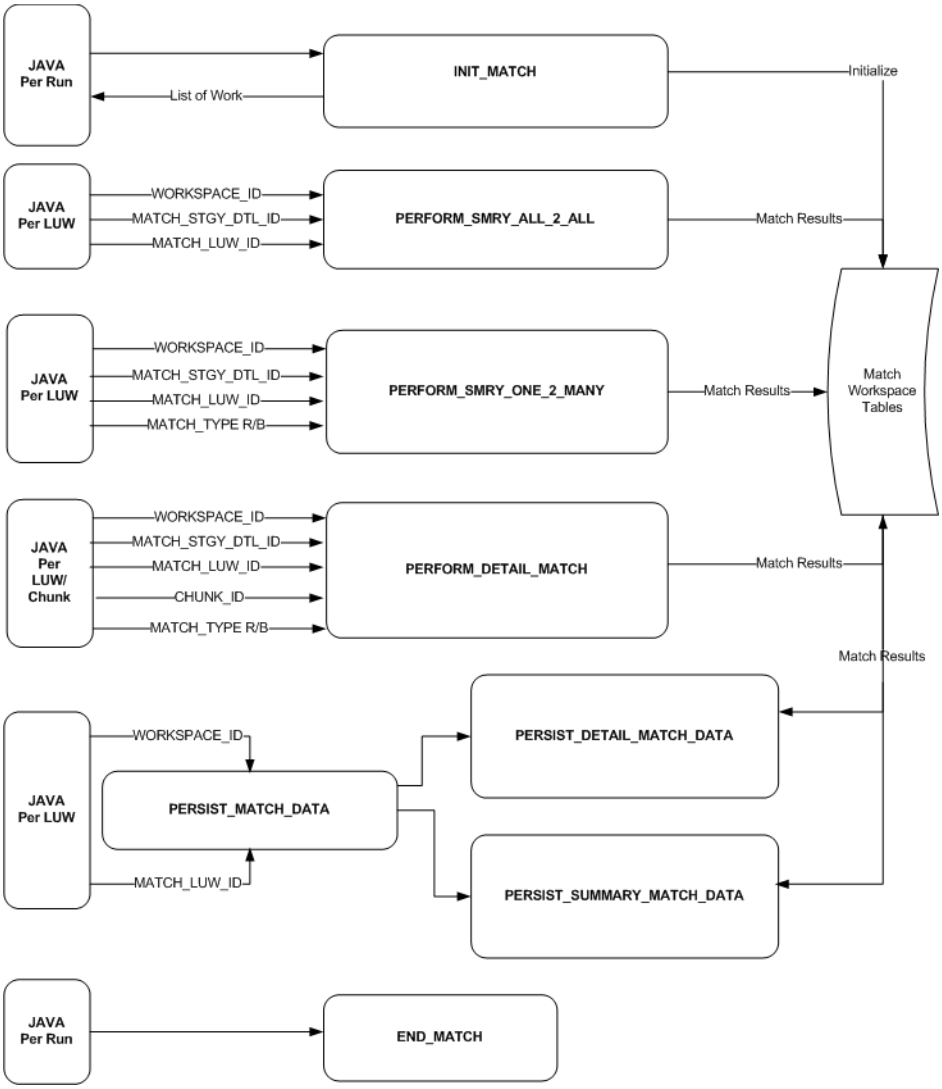
Assumptions and Scheduling Notes

Consider the following assumptions and scheduling notes.

- Auto-match cannot be run during the day when there are users online interacting with the system.
- Both the invoice unit cost and the unit cost of the PO must be expressed in the same currency. In order to compare the invoice unit costs with the PO's unit costs, auto-match does not engage in currency conversion. Match Keys which involve multiple currencies or vat regions or set of books will be removed from the matching process.

High-Level Flow Diagram

The following diagram offers a high-level view of the processing logic utilized within the auto-match batch process.



Primary Tables Involved

The following tables are involved in the Invoice Auto-Match batch process.

- IM_DOC_HEAD
- IM_INVOICE_DETAIL
- SHIPMENT(RMS)
- SHIPSKU(RMS)
- IM_PARTIALLY_MATCHED_RECEIPTS
- ORDHEAD(RMS)
- ORDSKU(RMS)
- ORDLOC(RMS)
- IM_TOLERANCE_LEVEL_MAP
- IM_SUPPLIER_OPTIONS
- IM_SYSTEM_OPTIONS

- IM_MATCH_INVC_WS
- IM_MATCH_INVC_DETL_WS
- IM_MATCH_RCPT_WS
- IM_MATCH_RCPT_DETL_WS
- IM_MATCH_GROUP_HEAD_WS
- IM_MATCH_GROUP_INVC_WS
- IM_MATCH_GROUP_RCPT_WS

Credit Note Auto-Match Batch Design

Credit Note Auto-Matching pairs credit note requests to corresponding credit notes sent by the supplier. The CreditNoteAutoMatchBatch attempts auto-matching of credit notes from suppliers, to credit note requests from the retailer without manual intervention. The batch also creates and resolves detail level discrepancies utilizing a predefined set of reason codes. These reason codes are defined within Invoice Matching through the System Options Maintenance screen. In addition, the batch utilizes a variety of configurable keys to allow for document groups to be matched in ways other than just distinct purchase order and location combinations.

When invoked, the batch creates a pool of matchable credit notes and credit note requests. The candidates are selected depending on which customizable fields are populated and a status of credit notes and credit note requests. For information, see the *Oracle Retail Invoice Matching User Guide*.

Once a pool of matchable documents is established, the batch proceeds to group the documents with respect to unique suppliers listed on the documents. Suppliers are the first layer of grouping, which facilitates further processing of each group in parallel using threads.

If threading is enabled for the batch, each supplier based group is processed in its own thread. Each supplier based group further divides the documents for that supplier into smaller document-key sets. These document-key sets are categorized by common attributes defined on the document itself. The attributes, also referred to as Configurable or Flexible 'Pool Keys' allow documents to be grouped in several combinations in addition to the distinct purchase order and location combination (which is the only combination possible in the current Invoice Auto-Matching framework).

Matching is not attempted for groups not containing both credit notes and credit note requests.

By default the CreditNoteAutoMatch process creates document-key sets based on the following key distinctions:

- Deal ID
- Deal Component ID
- Credit Note Request ID
- Original Invoice ID
- PO/Location combination

To enable the use of all five keys, the reference fields in the credit notes and credit note requests must be populated. For information, see the *Oracle Retail Invoice Matching User Guide*. The reference credit note request ID field holds the credit note request ID,

reference invoice ID field holds the original invoice ID, deal ID field holds the deal ID and deal detail ID field holds the deal component ID. In case none of these fields are populated with the required data, the PO/Location combination is the only key available to the CreditNoteAutoMatchBatch process.

Within each document-key set, matching is attempted using three algorithms: summary, one-to-one matching, and detail level matching. Summary-level matching attempts to match all credit notes with credit note requests at a summary level by comparing extended costs, or quantities within tolerance. One-to-one matching requires that extended costs or quantities of one distinct credit note match to only one distinct credit note request within tolerance. Line-level matching is only attempted if there is one unmatched credit note left. It attempts to match the line items of an unmatched credit note with line items of all unmatched credit note requests.

Below is the flow for attempting a match for each of the document-key set:

1. Summary Matching (matching algorithm)
2. One to One Matching (matching algorithm)
3. Line-level Matching (matching algorithm)

If Tax is enabled in the system, CreditNoteAutoMatchBatch only detects Tax discrepancies at the detail level. This means that when documents are being processed by the detail matching algorithm, a check is performed prior to matching, ensuring that the Tax codes and rates for each item on the credit note match those on the credit note request for the corresponding item. When a discrepancy is detected, processing for that document stops and detail matching is not performed for that document. In such a case, the Invoice Matching user will have to match and resolve the Tax discrepancy manually through the user interface.

Tolerances are handled in a manner similar to the Invoice auto-match batch process. The tolerances are first selected with respect to supplier, then with respect to the system. For information, see the *Oracle Retail Invoice Matching User Guide*.

If a match is achieved, the information related to the matched document is migrated to the history tables, and all CreditNoteAutoMatch Batch related tables are purged for those documents. The migration process is enabled depending on the value of the creditnoteautomatchbatch.workspace.cleanup property in the reim.properties file.

In case of an unsuccessful match manual intervention is required through online processes, and the match attempt related data for those documents is not cleaned up from the respective tables. See "Primary Tables Involved" in this section for more details on the tables involved.

Usage

The following arguments are applicable for the Credit Note Auto-Match Batch process:

```
reimcreditnoteautomatch batch-alias-name
```

Algorithms

The Credit NoteAutoMatch batch process includes the following algorithms.

- Summary Matching

  Credit notes and credit note requests in the document set are matched at the summary level by comparing extended costs. If the extended costs of the document set falls within tolerances, the documents are considered matched and flagged as such, processing continues with the next set. Note that since total

extended costs are being compared, only total merchandise amounts will be factored into the actual matching calculations. If the documents in the set are from a supplier that requires quantity matching, quantity matching will be performed within tolerances as well.

- One to One Matching

  One to one matching is a variation of summary matching. It requires that one distinct credit note matches to only one distinct credit note request within tolerance for the document set. Extended costs are compared and quantities are also compared if the supplier option for quantity matching is enabled.

- Detail Matching

  For a given document set, when only one credit note remains unmatched and multiple credit note requests remain unmatched, the system will attempt to match line items from the credit note to the credit note request at the line level. If a match is not found, discrepancies are created and routed for resolution. When discrepancies are created as part of the detail (line-level) matching process, they are automatically resolved by the batch process. This resolution will take place by selecting the appropriate pre-defined reason code from the system options and resolving the discrepancy. During the reason code action rollup process, these newly created resolution actions will be rolled up to create the appropriate resolution documents. In case no applicable reason codes exist in the system for the discrepancy, the credit note will not be matched and processing will stop for the document set.

Assumptions and Scheduling Notes

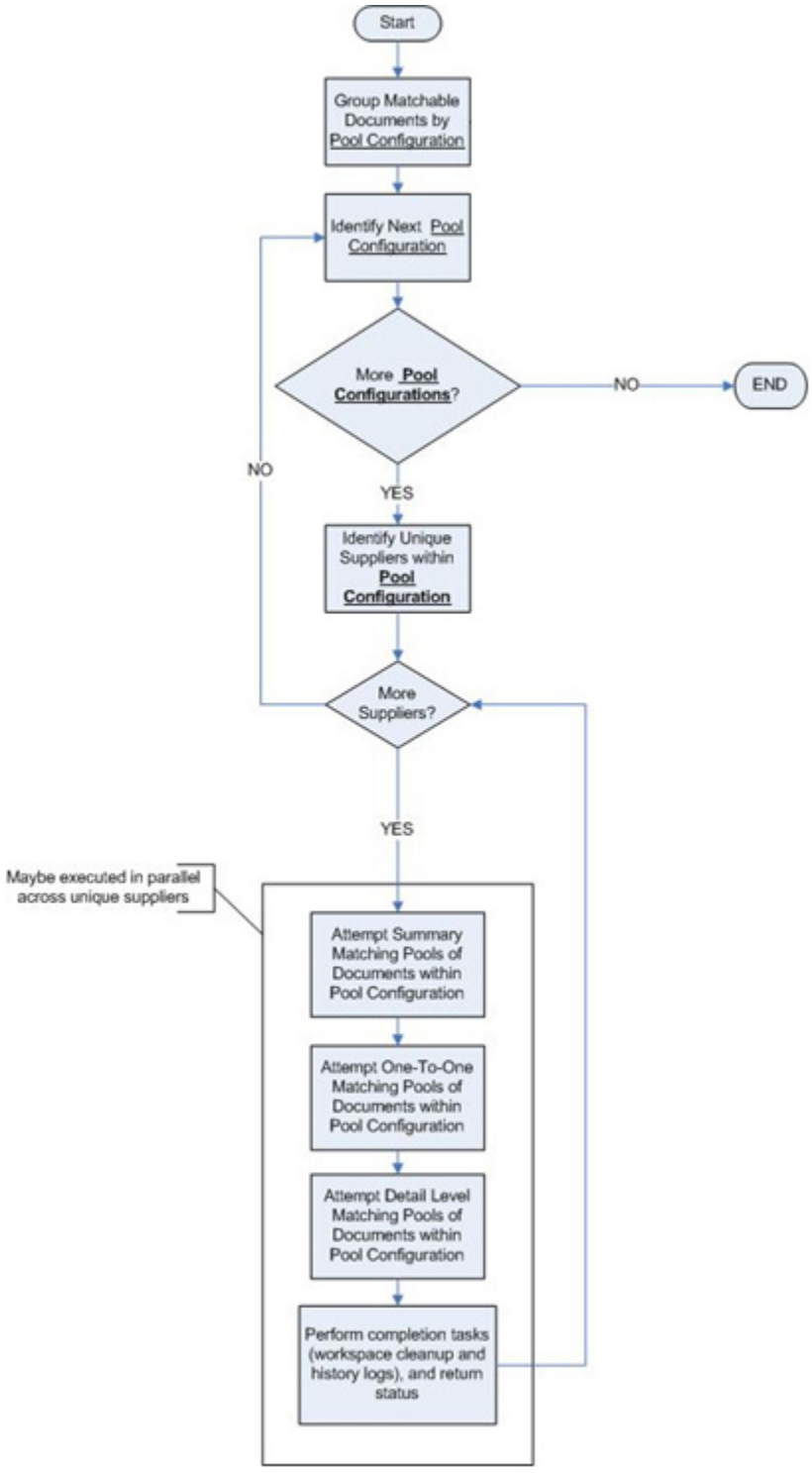
Consider the following assumptions and scheduling notes.

- Both the credit note and credit note request unit cost must be expressed in the same currency.
- The quantities on the credit note must be expressed in the same unit of measure as the quantities on the credit note requests. The batch performs no unit of measure conversion.

Post Processing

- CreditNoteAutoMatch updates the status of qualified documents that have been matched.
- The CreditNoteAutoMatch workspace is cleaned up depending on the related setting in the reim.properties file (refer to the Credit Note AutoMatch Workspace Cleanup Setting section in the reim.properties section).
- The batch creates and resolves discrepancies by utilizing pre-defined reason codes. The Reason Code Rollup Batch must ensure that the respective documents are created.

High-Level Flow Diagram



Primary Tables Involved

The following are lookup tables that must be populated.

Table 13–4

Table Name	Contents
IM_DOC_HEAD	Credit notes and credit note requests with relevant information (such as supplier and status).
IM_SUPPLIER_GROUP_ MEMBERS	Supplier group related information.
IM_DOC_DETAIL_ REASON_CODES	The Item Detail record for credit notes. Data related to items must exist in this table to enable line-level matching.
IM_TOLERANCE_LEVEL_ MAP	Tolerance properties associated with supplier. The data is required when performing matches within tolerances.
IM_SYSTEM OPTIONS	Properties associated with the Invoice Matching function, such as enabling TAX or enabling tolerances.

The following are tables to which the process posts data.

Table 13–5

Table Name	Contents
IM_MATCH_POOL_ CONFIG	Data for the matching process. This data determines which groupings the system utilizes when attempting to match and also dictates the order in which the groupings run.
IM_MATCH_DOC	The pool of documents that the batch process will attempt to match.
IM_MATCH_POOL_ TOLERANCES	The calculated tolerances for each candidate document to be matched.
IM_MATCH_POOL_ RESULTS	Cost and quantity total for a document set being matched and the variance between the documents being matched. Also included is the party the variance favors (retailer or supplier).
IM_MATCH_POOL_ITEM	Actual item detail unit cost and quantities to be used for matching. Details may be from IM_DOC_DETAIL_REASON_ CODES or IM_INVOICE_DETAILS, depending on the type of match performed.
IM_MATCH_QTY_VAR	The quantity discrepancy calculated while attempting a match in a document set.
IM_MATCH_COST_VAR	The cost discrepancy calculated while attempting a match in a document set.

The following new history tables are populated upon the successful completion of the CreditNoteAutoMatch batch. The tables allow the retailer to track match history and locate aggregate data in the other match history tables based on the appropriate match and document type.

Table 13–6

Table Name	Contents
IM_MATCH_DOC_HIST	Upon successful completion of the matching process, documents contained in IM_MATCH_DOC are moved to this history table.
IM_MATCH_POOL_ITEM_ HIST	History of the items that were on the credit note when matched.
IM_MATCH_POOL_ RESULTS_HIST	Data from the MATCH_POOL_RESULTS table is moved to this table after a successful match.

Table 13–6 (Cont.)

Table Name	Contents
IM_MATCH_QTY_VAR_HIST	
IM_MATCH_COST_VAR_HIST	History related to any quantity or cost variance detected during the match process.

The following tables are populated for compatibility with the existing Invoice Matching history maintenance data model.

- IM_CN_SUMMARY_MATCH_HIS
- IM_CN_DETAIL_MATCH_HIS

Receipt Write-Off Batch Design

Retailers track received goods that are not invoiced, and they must have the ability to 'write-off' these goods for financial tracking. Two types of processes can determine when these written-off goods will be written to financials: purged receipts from merchandising system, and close open receipts from invoice matching. Because receipts can be purged outside of the invoice matching dialogue, these purged receipts must be maintained until their unmatched amount has been accounted for. These receipts are tracked through STAGE_PURGED_SHIPMENTS and STAGE_PURGED_SHIPSKUS. Every purged shipment record that is not fully matched will have a record by item written to the stage tables. In addition, invoice matching has a system parameter (which can be overwritten at the supplier level) defining the maximum amount of time an open, non-fully matched receipt will be available for matching.

Every time the write-off process is run, each non-fully matched open receipt received date is compared with the current date minus the system parameter. If the received date is before this difference, then the receipt will be written-off and the invoice match status is closed.

The department/class of each receipt item must be identified to ensure accurate accounting. The form of the accounting distribution is as follows:

Table 13–7

Transaction Type	Sign	Value	Notes
Unmatched receipt	Debit	Value of unmatched items on receipt	
Receipt write-Off	Credit	Same as above	
Trade accounts payable	Credit	0	Written as a matter of form

This account distribution mapping is set up through the account cross-reference screen.

Note: If IM_SUPPLIER_OPTIONS.CLOSE_OPEN_RECEIPT_MONTHS is not defined, the value is retrieved from IM_SYSTEM_OPTIONS.CLOSE_OPEN_RECEIPT_MONTHS.

Usage

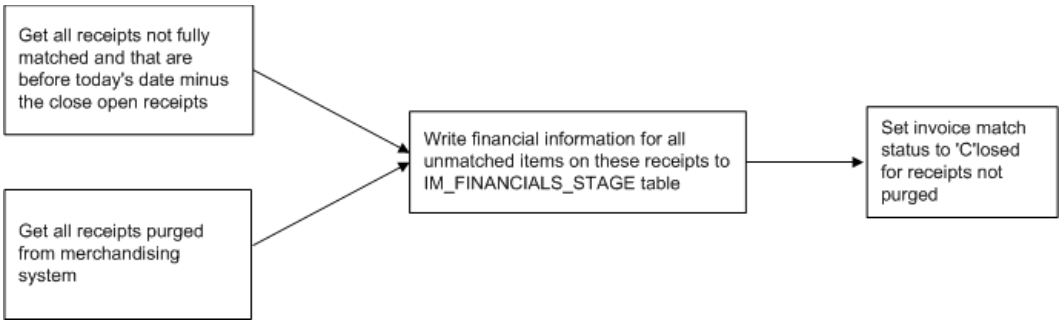
The following arguments are applicable for the Receipt Write-Off Batch process:

```
reimreceiptwriteoff batch-alias-name
```

Assumptions and Scheduling Notes

- When setting up the Close Open Receipt Months in ReIM Supplier Options and/or System Options, the value should be less than or equal to RMS UNIT_OPTIONS.ORDER_HISTORY_MONTHS if the intention is to have invoice matching pick up receipts prior to purging.
- Auto-match and any associated processing must be run prior to this batch processing.

High-Level Flow Diagram



Primary Tables Involved

The following tables are involved in the Receipt Write-off batch process.

REIM

- IM_FINANCIALS_STAGE
- IM_SYSTEM_OPTION
- IM_SUPPLIER_OPTIONS
- IM_PARTIALLY_MATCHED_RECEIPTS

RMS

- UNIT_OPTIONS
- SHIPMENT
- STAGE_PURGED_SHIPMENT
- SHIPSKU
- STAGE_PURGE_SHIPSKU

Reason Code Action Rollup Batch Design

Reason code actions are resolutions assigned at the discrepancy line level. A number of fixed actions are available to resolve a line item discrepancy; the specific results depend on the action.

The resolution posting process sweeps the IM_RESOLUTION_ACTION table and creates debit and credit memos as needed. Only a single debit or credit memo is created per invoice/discrepancy type, with line details from all related actions for the same discrepancy type.

This process does not delete these records when completed; rather, they are deleted after posting.

The action staging table is used during posting to post the reason code actions to the financial staging table.

Usage

The following arguments are applicable for the Reason Code Action Rollup Batch process:

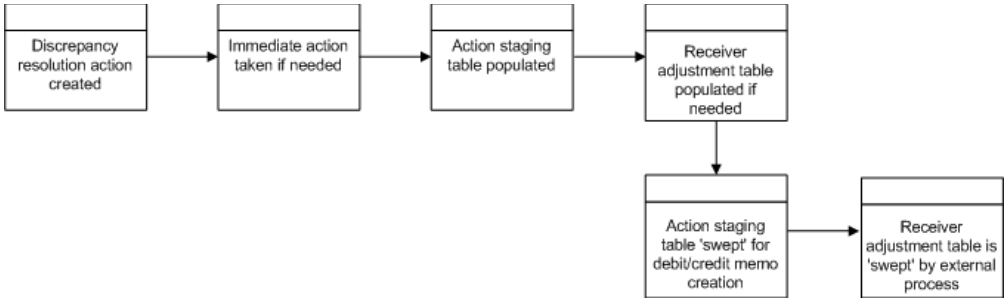
```
reimrollup batch-alias-name
```

Assumptions and Scheduling Notes

The memo staging table sweep must occur before the posting batch process, or a delay of one day results before posting can occur.

High-Level Flow Diagram

The following diagram offers a high-level view of the processing logic utilized within the reason code action rollup batch process.



Primary Tables Involved

The following tables are involved in the Reason Code Action Rollup batch process.

- IM_DOC_HEAD
- IM_INVOICE_DETAIL
- IM_PARTIALLY_MATCHED_RECEIPTS
- IM_RESOLUTION_ACTION
- IM_RECEIVER_COST_ADJUST
- IM_RECEIVER_UNIT_ADJUST

Financial Posting Batch Design

For each invoice, the batch process writes applicable financial accounting transactions to either of the following tables:

- The Financials staging table, IM_FINANCIALS_STAGE.
- The AP staging tables, IM_AP_STAGE_HEADER and IM_AP_STAGE_DETAIL, or the IM_FINANCIALS_STAGE, depending on the transaction type.

The processing occurs after discrepancies for documents have been resolved by resolution documents. Once all of the resolution documents for a matched invoice are built, and all of the RCA/RUA external processing has been confirmed, the process inserts financial accounting transactions to the financials staging table, to represent the resolution and consequent posting of the invoice. The process also inserts financial accounting transactions for the approved documents that are being handled.

Once all of the transactions have been written, the process switches the status of the current invoices/documents to Posted and moves on to the next invoice/document.

If a segment look-up fails, the failed record is written to a financials error table.

Usage

The following arguments are applicable for the Financial Posting Batch process:

```
reimposting batch-alias-name
```

Assumptions and Scheduling Notes

Before posting can occur, the following information must be set up:

- Segment definitions in the GL options.
- GL account segments on the GL Options screen.
- All the accounts using the GL Cross Reference screen.
- Country
- Location
- Dept
- Class

If dynamic segments are defined, the values for the segments must be defined in the applicable tables, IM_DYNAMIC_SEGMENT_DEPT_CLASS or IM_DYNAMIC_SEGMENT_LOC.

Primary Tables Involved

The following tables are involved in the Financial Posting batch process.

- The IM_DOC_HEAD table contains the matched, void, and approved documents.
- The IM_DOC_NON_MERCH table contains the non-merchandise costs for invoices.

Lookup Tables that must be Populated

- IM_GL_OPTIONS. Order of segments, business attributes, and dynamic segments defined.
- IM_GL_CROSS_REF. Account values defined for account types and account codes.
- IM_DYNAMIC_SEGMENT_DEPT_CLASS. Accounts defined for each department/class combination.

- IM_DYNAMIC_SEGMENT_LOC. Accounts defined for each location/company combination.

Tables to Which the Process Posts Data

Note: The table to which the process posts data is either IM_FINANCIALS_STAGE or IM_AP_STAGE_HEAD

IM_FINANCIALS_STAGE

- Transaction code
- Debit/credit indicator
- Invoice ID
- Invoice date
- Supplier
- Purchase order (if available)
- Shipment/receipt (only if unmatched receipt is being written)
- Currency
- Amount
- Best terms ID
- Terms date
- Pre-paid indicator
- Comments
- Create user ID
- Create date-time
- Segments that determine the mapping account in the external financial system (as defined in the IM_GL_CROSS_REF table).

IM_AP_STAGE_HEAD

- Sequence Number: Automatically generated line numbers 1, 2, 3, and so on; incremented for each detail record per DOC ID for identification purposes.
- Doc_id: Similar to IM_FINANCIALS_STAGE.
- The Invoice Type Lookup Code for merchandise invoices and credit memos (where IM_DOC_HEAD.TYPE is MRCHI, CRDMEC or CRDMEQ) is STANDARD. For positive non-merchandise invoices (where IM_DOC_HEAD.TYPE is NMRCHI) the Invoice Type Lookup Code also is Standard. For negative non-merchandise invoices and all other documents, the Invoice Lookup Code is CREDIT.
- invoice_number: The concatenated data is as follows:
  - chars 1-34: the first 34 characters from the EXT DOC ID
  - char 35: a hyphen
  - chars 36-50: the DOC ID
- Vendor: Same as for current im staging table.

- Oracle_site_id:
  - The loc from this transaction to read new RMS Location/Org Unit data to find the Org Unit.
  - The Org Unit to read new RMS Supplier Addr/Org Unit/Site ID data to find Oracle Site ID.
  - The Org Unit of the Location from this transaction should match the Org Unit of the Site ID. Otherwise, this field value will be null.
- Currency Code: Valued if this is a foreign currency invoice, otherwise null.
- Exchange Rate: If exchange rate is valued, this should be the literal, USER; otherwise blank.
- Exchange Rate Type
- Document Date: Same as in current im staging table.
- Amount: The TOTAL amount including tax.
- Best Terms Date: Same as in current im staging table.
- Segment1: Same as in current IM financials staging table.
- Segment2: Same as in current IM financials staging table.
- Segment3: Same as in current IM financials staging table.
- Segment 4: Same as in current IM financials staging table.
- Segment 5: Same as in current IM financials staging table.
- Segment 6: Same as in current IM financials staging table.
- Segment 7: Same as in current IM financials staging table.
- Segment 8: Same as in current IM financials staging table.
- Segment 9: Same as in current IM financials staging table.
- Segment 10: Same as in current IM financials staging table.
- Create Date: Same as in current IM financials staging table.
- Best Terms ID: Same as in current IM financials staging table.

IM_AP_STAGE_DETAIL

- Doc_id
- Sequence number: Automatically generated line numbers 1, 2, 3, and so on; incremented for each detail record per DOC ID; for identification purpose.
- Transaction Code
- Line Type Lookup Code: This value varies. The rules are:
  - If the tran-code is UNR or VWT or REASON or CRN then this value is ITEM.
  - If this is a generated tax line, then this value will be TAX.

    Recent modifications have been made to the ReIM posting process to better support integration with EBS with respect to VAT requirements. Previous to the modifications, ReIM would post Invoices to the staging tables which passed information to Accounts Payables in a manner where in certain scenarios, the VAT lines could not be easily associated with corresponding Merchandise lines on the invoice. The details of the association between Items

and VAT Codes/Rates is available in ReIM, however it could be lost during the posting process with Accounts Payable.

These new modifications provided a more detailed breakdown of information for items by VAT Code and the association with the appropriate VAT lines. Previous to the modifications, ReIM made it's postings for financial integration by rolling up the Items in the posting to a GL Account Segment level. So, all RMS items that are mapped to the same GL Account Codes in ReIM will be combined into a single posting line. Along with this, TAX lines for the item lines are also posted, one for each VAT Rate that was applicable to the items included in the ITEM line. This did not provide the ability to easily determine the VAT basis that was used to determine the VAT line once posted to the financial system.

The modification changes the level at which the Item lines are posted so that the VAT Rate of the items provides a further breakdown of the Item line posting. The posting now makes the same roll up to the common GL Accounts Segment level and then provides a further breakdown to the VAT code level. TAX line are then posted with each Item line for the corresponding VAT rate of the items. No Item line would have more than 1 TAX line associated with it.

  - If none of the above, then this value will be MISCELLANEOUS.
- Amount
- Vat Code: Same as in current IM staging table except for generated tax lines, where the amount for this line should be the amount from the taxable line times the tax rate
- Segment1: For regular lines, same as in current staging table; for generated tax line, use values from source line.
- Segment2: (see rules for segment 1)
- Segment3: (see rules for segment 1)
- Segment4: (see rules for segment 1)
- Segment5: (see rules for segment 1)
- Segment6: (see rules for segment 1)
- Segment7: (see rules for segment 1)
- Segment8: (see rules for segment 1)
- Segment9: (see rules for segment 1)
- Segment10: (see rules for segment 1)
- Create Date: Same as in current IM staging table.

EDI Invoice Download Batch Design

The EDI invoice download process retrieves debit memos, credit note requests, and credit memos in 'approved' or 'posted' status from the resolution posting process and creates a flat file. The client converts the flat file into an EDI format and sends it through the EDI invoice download transaction set to the respective vendors.

Usage

The following arguments are applicable for the EDI Invoice Download Batch process:

```
reimediinvdownload batch-alias-name
```

Assumptions and Scheduling Notes

Consider the following assumptions and scheduling notes.

- All data is valid in the IM_DOC_HEAD tables. ReIM does not validate details.
- Auto-match must run prior to the EDI invoice download.

Primary Tables Involved

The EDI invoice download batch process reads from the following tables:

- IM_DOC_HEAD
- IM_DOC_DETAIL_REASON_CODES
- IM_DOC_NON_MERCH
- IM_DOC_DETAIL_COMMENTS

Restart and Recovery

If the EDI invoice download aborts while processing, an incomplete file is generated. To generate a complete file, the process simply needs to be rerun and allowed to fully process. If the cause of the aborted process is software related, this action might not solve the issue; other steps may be required to ensure that the process completes its entire initial run.

Complex Deal Upload Batch Design

The Complex Deal Upload batch process reads data from header and detail complex deals staging tables in RMS.

For each combination of deal ID and deal detail ID on the RMS staging tables, the batch process creates a credit memo, a debit memo, or a credit note request, depending upon an indicator on the staging tables.

The batch process also copies most of the data from the RMS staging tables into one ReIM detail table (IM_COMPLEX_DEAL_DETAIL). This data is later referenced during the posting process for the created documents.

Usage

The following arguments are applicable for the Complex Deal Upload Batch process:

```
reimcomplexdealupload batch-alias-name block-size
```

Primary Tables Involved

Note: For descriptions of RMS tables, see the *Oracle Retail Merchandising System Data Model*.

- STAGE_COMPLEX_DEAL_HEAD (RMS table)
- STAGE_COMPLEX_DEAL_DETAIL (RMS table)

- IM_DOC_HEAD. This table holds general information for documents of all types. Documents include merchandise invoices, non-merchandise invoices, consignment invoices, credit notes, credit note requests, credit memos, and debit memos. Documents remain on this table for SYSTEM_OPTIONS.DOC_HISTORY_MONTHS after they are posted to the ledger.
- IM_DOC_DETAIL_REASON_CODES. This table contains quantity/unit cost adjustments for a given document/item/reason code.
- IM_DOC_TAX. This table associates the document with its value added tax information.
- IM_COMPLEX_DEAL_DETAIL. This table holds the details of the complex deal stored in ReIM. It is used during complex deal detail posting.
- IM_COMPLEX_DEAL_DETAIL_TAX. This table holds the tax information of the complex deal.

Multi-Threading

The Complex Deals upload batch is run in multi-threaded mode as follows:

- reimcomplexdealupload user/password BlockSize PartitionNo

BlockSize

The BlockSize is used to decide how many deal IDs to process in every thread. It should be greater than 1.

For example, if there are 15 deals to be processed in the staging tables and BlockSize input argument is provided as 3, then there will be 5 threads to process 3 deals each simultaneously.

A total of 3 deals records are processed in each of the 5 threads.

Generation of Debit Memo (or Credit Note Requests) for Deals

The RMS system generates Debit Memos (or Credit Note Requests) for Fixed or Complex deals and pass them through to ReIM via custom upload batch programs. The RMS system includes a system option called 'Credit Memo Level' which will control the level at which the Debit Memo (or Credit Note Request) is generated.

The valid values for the Credit Memo Level option are:

- L – Location
- T – Transfer Entity
- B – Set of Books
- D – Deal/Component

Therefore, if a retailer wishes to generate a separate Debit Memo (or Credit Note Request) for each location on a deal, they would set the Credit Memo Level option to 'L', and RMS would send a separate transaction for each location on the deal. If this level of detail is not needed, the retailer could set the option to 'D' and only a single document would be sent to ReIM for each deal/component. Note that if the deal/component level is selected, ReIM still internally tracks the locations on each deal and will use this information to credit the correct locations on the deal when making the accounting entries.

Fixed Deal Upload Batch Design

The Fixed Deal Upload batch process reads data from header and detail fixed deals staging tables in RMS.

For each deal ID on the RMS staging tables, the batch process creates a credit memo, a debit memo, or a credit note request, depending upon an indicator on the staging tables.

The batch process also copies most of the data from the RMS staging tables into one ReIM detail table (IM_FIXED_DEAL_DETAIL). This data is later referenced during the posting process for the created documents.

For non-merchandise fixed deals that are not associated with an RMS location, the org unit has been added to the RMS staging table. During the Fixed Deal upload process, the set of books ID associated with this org unit is used to access a new table (FIXED_DEAL_SOB_LOC_DEFAULT) to get the location to use for the deal document in IM_DOC_HEAD. Then, the resolution posting job populates the financial staging tables with the set of books ID associated with the location just like it does with all other documents.

Usage

The following arguments are applicable for the Fixed Deal Upload Batch process:

```
reimfixeddealupload batch-alias-name block-size partition-no partition-size
```

Primary Tables Involved

Note: For descriptions of RMS tables, see the *Oracle Retail Merchandising System Data Model*.

- STAGE_FIXED_DEAL_HEAD (RMS table)
- STAGE_FIXED_DEAL_DETAIL (RMS table)
- IM_DOC_HEAD. This table holds general information for documents of all types. Documents include merchandise invoices, non-merchandise invoices, consignment invoices, credit notes, credit note requests, credit memos, and debit memos. Documents remain on this table for SYSTEM_OPTIONS.DOC_HISTORY_MONTHS after they are posted to the ledger.
- IM_DOC_NON_MERCH. This table holds various user-defined non-merchandise costs associated with an invoice. Non merchandise costs can be associated with merchandise invoice if the IM_SUPPLIER_OPTIONS.MIX_MERCH_NON_MERCH_IND for the vendor is 'Y'. If the MIX_MERCH_NON_MERCH_IND for the vendor is N, non merchandise expenses can only be on non merchandise invoice documents.
- IM_DOC_TAX. This table associates the document with its value added tax information.
- IM_FIXED_DEAL_DETAIL. This table holds the details of the fixed deals in the ReIM system. It will be used during fixed deal detail posting.
- IM_FIXED_DEAL_DETAIL_TAX. This table holds the tax information of the fixed deal.

Multi-Threading

The Fixed Deals upload batch is run in multi-threaded mode as follows:

- reimfixeddealupload user/password BlockSize PartitionNo

BlockSize

The BlockSize is used to decide how many deal IDs to process in every thread. It should be greater than 1.

For example, if there are 15 deals to be processed in the staging tables and BlockSize input argument is provided as 3, then there will be 5 threads to process 3 deals each simultaneously.

A total of 3 deals records are processed in each of the 5 threads.

PartitionNo

The PartitionNo is used by huge data block that are in the units of millions (for example, 4 million).

The batch is used by the query to pick all the records and it retrieves ALL the deal numbers to be processed by the batch.

For example, the input command line arguments:

reimfixeddealupload user/password 3 1

Generation of Debit Memo (or Credit Note Requests) for Deals

The RMS system generates Debit Memos (or Credit Note Requests) for Fixed or Complex deals and pass them through to ReIM via custom upload batch programs. The RMS system includes a system option called 'Credit Memo Level' which will control the level at which the Debit Memo (or Credit Note Request) is generated.

The valid values for the Credit Memo Level option are:

- L – Location
- T – Transfer Entity
- B – Set of Books
- D – Deal/Component

Therefore, if a retailer wishes to generate a separate Debit Memo (or Credit Note Request) for each location on a deal, they would set the Credit Memo Level option to 'L', and RMS would send a separate transaction for each location on the deal. If this level of detail is not needed, the retailer could set the option to 'D' and only a single document would be sent to ReIM for each deal/component. Note that if the deal/component level is selected, ReIM still internally tracks the locations on each deal and will use this information to credit the correct locations on the deal when making the accounting entries.